FUJITSU

SI3603FSG-01E

F²MC-8FX Family

SOFTUNE First Step Guide

MB2146-09 BGM Adapter

SOFTUNE First Step Guide
11 June 2004

1. The contents of this document are subject to change without notice. Customers are advised to consult with FUJITSU sales representatives before ordering.

2. The information, such as descriptions of function and application circuit examples, in this document are presented solely for the purpose of reference to show examples of operations and uses of Fujitsu semiconductor device; Fujitsu does not warrant proper operation of the device with respect to use based on such information. When you develop equipment incorporating the device based on such information, you must assume any responsibility arising out of such use of the information. Fujitsu assumes no liability for any damages whatsoever arising out of the use of the information.

3. Any information in this document, including descriptions of function and schematic diagrams, shall not be construed as license of the use or exercise of any intellectual property right, such as patent right or copyright, or any other right of Fujitsu or any third party or does Fujitsu warrant non-infringement of any third-party' s intellectual property right or other right by using such information. Fujitsu assumes no liability for any infringement of the intellectual property rights or other rights of third parties which would result from the use of information contained herein.

4. The products described in this document are designed, developed and manufactured as contemplated for general use, including without limitation, ordinary industrial use, general office use, personal use, and household use, but are not designed, developed and manufactured as contemplated (1) for use accompanying fatal risks or dangers that, unless extremely high safety is secured, could have a serious effect to the public, and could lead directly to death, personal injury, severe physical damage or other loss (i.e., nuclear reaction control in nuclear facility, aircraft flight control, air traffic control, mass transport control, medical life support system, missile launch control in weapon system), or (2) for use requiring extremely high reliability (i.e., submersible repeater and artificial satellite). Please note that Fujitsu will not be liable against you and/or any third party for any claims or damages arising in connection with above-mentioned uses of the products.

5. Any semiconductor devices have an inherent chance of failure. You must protect against injury, damage or loss from such failures by incorporating safety design measures into your facility and equipment such as redundancy, fire protection, and prevention of over-current levels and other abnormal operating conditions.

6. If any products described in this document represent goods or technologies subject to certain restrictions on export under the Foreign Exchange and Foreign Trade Law of Japan, the prior authorization by Japanese government will be required for export of those products from Japan.

CONTENTS

PREFACE

■ Purpose of Manual

This manual describes the procedures for developing F²MC-8FX software using F²MC-8L/8FX Family ***SOFTUNE*** Professional Pack (***SOFTUNE***) and its functions.
This manual is intended to give a quick understanding of how to operate ***SOFTUNE*** efficiently.

■ Readers

This manual is written for engineers who develop software using ***SOFTUNE***.

■ Features of Manual

This manual introduces the procedure for software development with ***SOFTUNE*** using simple samples.
It also describes the useful functions and technical knowledge of ***SOFTUNE***.

■ Versions of Tools

The versions of the tools used in this manual are as follows:

F²MC-8L/8FX Family ***SOFTUNE*** Professional Pack REV: 300012

- SOFTUNE Workbench V30L29
- SOFTUNE C Compiler V30L09
- SOFTUNE Assembler Pack REV: 300011

Note: ***SOFTUNE*** includes C Checker and C Analyzer, but this manual does not describe them. For more information about these tools, refer to the respective ***manuals***. The MB2146-09 is assumed as the In-Circuit Emulator (ICE).

■ Cautions

- Use this ***manual*** as a part of the reference materials for various manuals for ***SOFTUNE***.
- Use the included sample programs after evaluating them.
- Viewing the included information requires **Adobe Acrobat Reader 5.0**.

■ 1. GENERAL

This chapter describes the functions and features of ***SOFTUNE***.

■ 2. INSTALLING

This chapter explains how to install ***SOFTUNE***.

■ 3. DEVELOPMENT PROCEDURE

This chapter covers the procedure for software development using ***SOFTUNE***.

■ 4. TUTORIAL

This chapter gives examples of software development using the F²MC-8L/8FX family ***SOFTUNE*** V3, including Project creation and Debugger operation through the use of simple samples.

■ 5. USEFUL FUNCTIONS

This chapter describes the useful functions of ***SOFTUNE*** Workbench.

■ Appendix FAQ

Appendix A-C lists frequently asked questions and technical knowledge in question-and-answer form.

■ Related manuals

When using this system, please also refer to the following manuals.

- F²MC-8L/8FX Family ***SOFTUNE*** Workbench OPERATION MANUAL
- F²MC-8L/8FX Family ***SOFTUNE*** Workbench USER'S MANUAL
- F²MC-8L/8FX Family ***SOFTUNE*** Workbench COMMAND REFERENCE MANUAL
- F²MC-8L/8FX Family ***SOFTUNE*** C COMPILER MANUAL
- F²MC-8L/8FX Family ***SOFTUNE*** ASSEMBLER MANUAL
- F²MC-8L/8FX Family ***SOFTUNE*** LINKAGE KIT MANUAL
- F²MC-8L/8FX Family ***SOFTUNE*** C CHECKER MANUAL
- F²MC-8L/8FX Family ***SOFTUNE*** C ANALYZER MANUAL

1. GENERAL

This chapter describes the features of ***SOFTUNE*** as an Integrated Development Environment and the features of project management functions.

1.1 Features as Integrated Development Environment

■ Manager Function

This function supports operations, such as coding, debugging and compiling, performed in software development comprehensively.
It enables the user to develop software without paying attention to starting up language tools including a Compiler and Debugger, and making option setting files.

■ Debugger Function

The user can use two Debuggers (Simulator Debugger, Emulator Debugger) as needed.
The user can easily set the environments, such as the debugging environment and operating environment of the MCU, etc.

■ Function for Checking and Analyzing Source Files

SOFTUNE has C Checker and C Analyzer tools to check and analyze source files written in C. For details, refer to the manuals for the respective tools.

■ Setting Function for Setting Editors and Tools

Although ***SOFTUNE*** has an editor, the user can register any editor and use it.
The user can also set any execution file and execute it from the ***SOFTUNE*** Workbench desktop.
For details, refer to Help (***Section 4.7 Setup of SOFTUNE Workbench Operation Manual***).

One Point !

= Debugger type =

Debugger type	Equipment required for program evaluation
Simulator Debugger	Personal computer
Emulator Debugger	Personal computer, ICE, target board

- Simulator Debugger

  The Simulator Debugger simulates the MCU operations (instruction execution, memory space, I/O ports, interrupts, reset, low-power consumption modes, etc.) only with software and evaluates the program.

  The Simulator Debugger is used to evaluate the uncompleted evaluation system and check a single program operation.

  The Simulator Debugger does not support the internal resources and registers associated with them not mentioned in the ***SOFTUNE*** Workbench 'USER'S MANUAL – Simulator Debugger'.

- Emulator Debugger

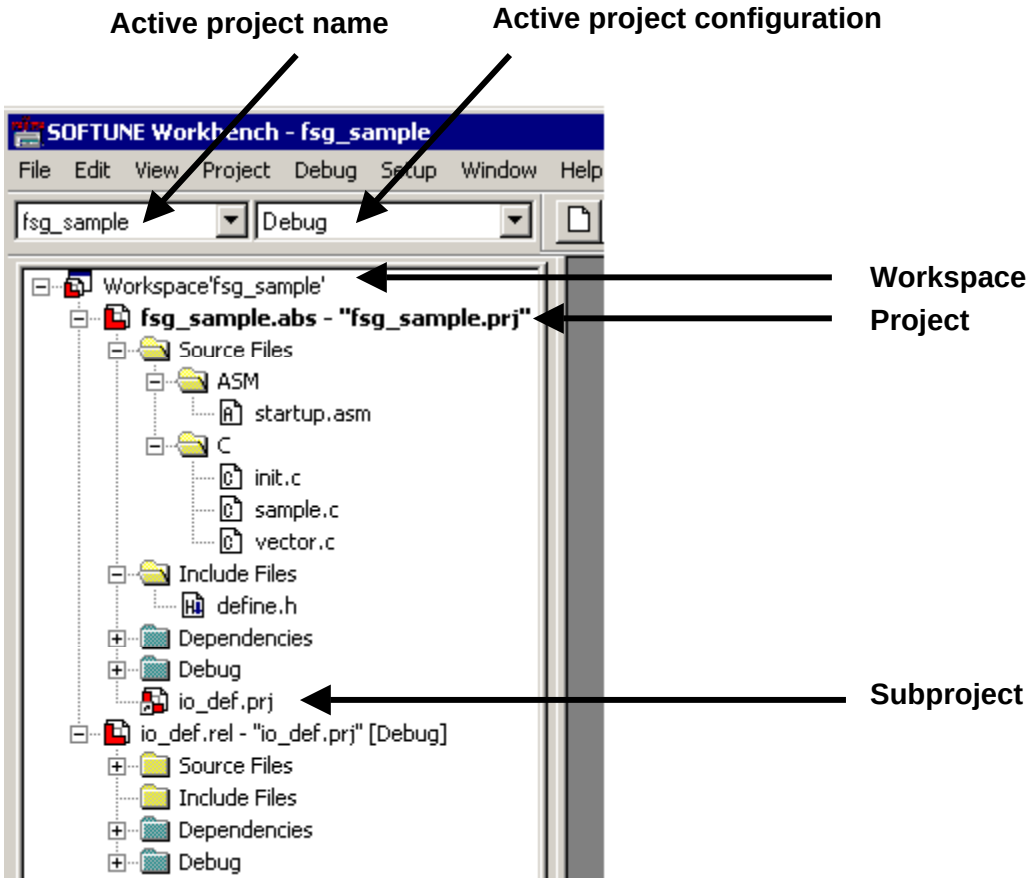
  The Emulator Debugger is software that controls the in-circuit Emulator (ICE) through a communication line (USB) from the host computer and evaluates the program.

1.2 Features of Project Management Functions

■ Features of Project Management Functions

The project management functions of F²MC-8L/8FX Family ***SOFTUNE*** Workbench are briefly described below.

- Introduction of a workspace allows registration and hierarchical management of multiple projects.
- Making a subproject enables definition of dependencies between projects.
- Making a project configuration allows setting, saving, and selection of two or more combinations of options in one project.
- Making any folder in the Project enables hierarchical management of Member files.
- Folders or Member files can be added and moved by dragging and dropping.



= Explanation of Terms =

- **Workspace**

A workspace is a unit to register and manage two or more projects. Firstly making a workspace allows the user to add different versions of projects. By adding and hierarchizig these projects, these can have dependencies one another.

- **Project**

A project is a basic unit that manages an operating environment of application programs to be developed. Project information includes source file, tool options to be used, and setting information of an debugger, etc. Workbench saves these information in a project file.

- **Active project**

An active project is a project for which menu operations, such as [Make], [Build], [Compile/Assemble], [Start Debug], or [Update Dependencies], will be executed. A subproject depends on the active project. If projects are in a workspace, one active project always exits.

- **Subproject**

A subproject is a project that is dependent on other projects. Defining a subproject allows it to be Made or Built to link its target file before the parent project is Made or Build.

- **Project configuration**

Project configuration is setting of projects or configuration of Members that provides different build methods for the target file in the same project. Operations, such as Make, Build, or Compile/Assemble, are executed in units of project configurations. When a new project is made, the project configuration is made under a default name of **Debug**. When a project configuration is added, the setting of the project configuration selected at its source is applied. Using the project configuration, the user can make multiple projects settings with different language tool options and Member information, such as for debugging, release, and trial, within one project.

2. INSTALLING

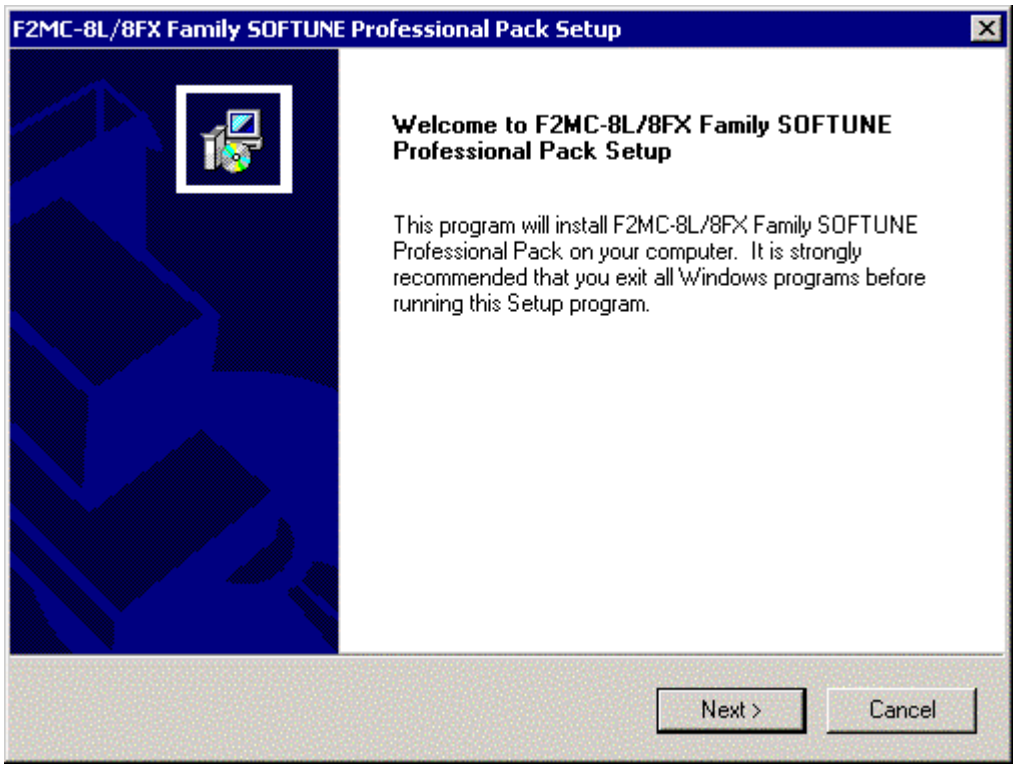
■ Procedure

This chapter explains how to install ***SOFTUNE*** using an example of installation of F^2MC-8L/8FX Family ***SOFTUNE*** Professional Pack.

《 Setup procedure 》

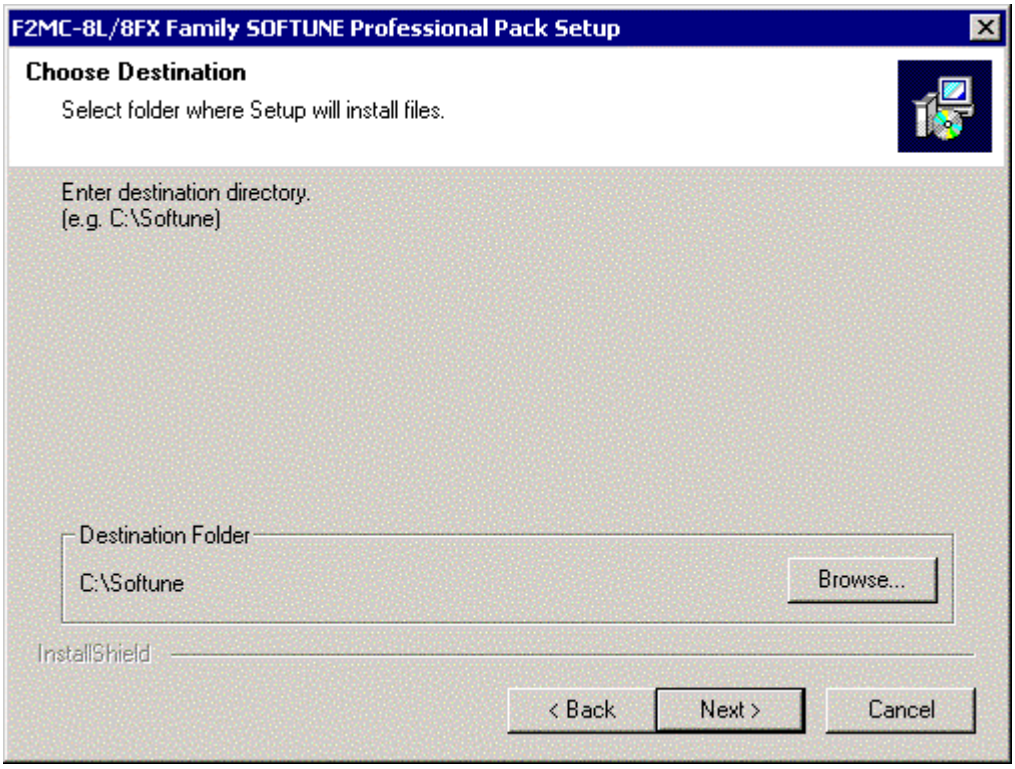
(1) Insert the CD-ROM into the drive.

(2) When setup.exe starts up, the Setup menu appears.



(3) Specify the destination directory.

See ■ ***Cautions*** before specifying the directory.



■ **Cautions**

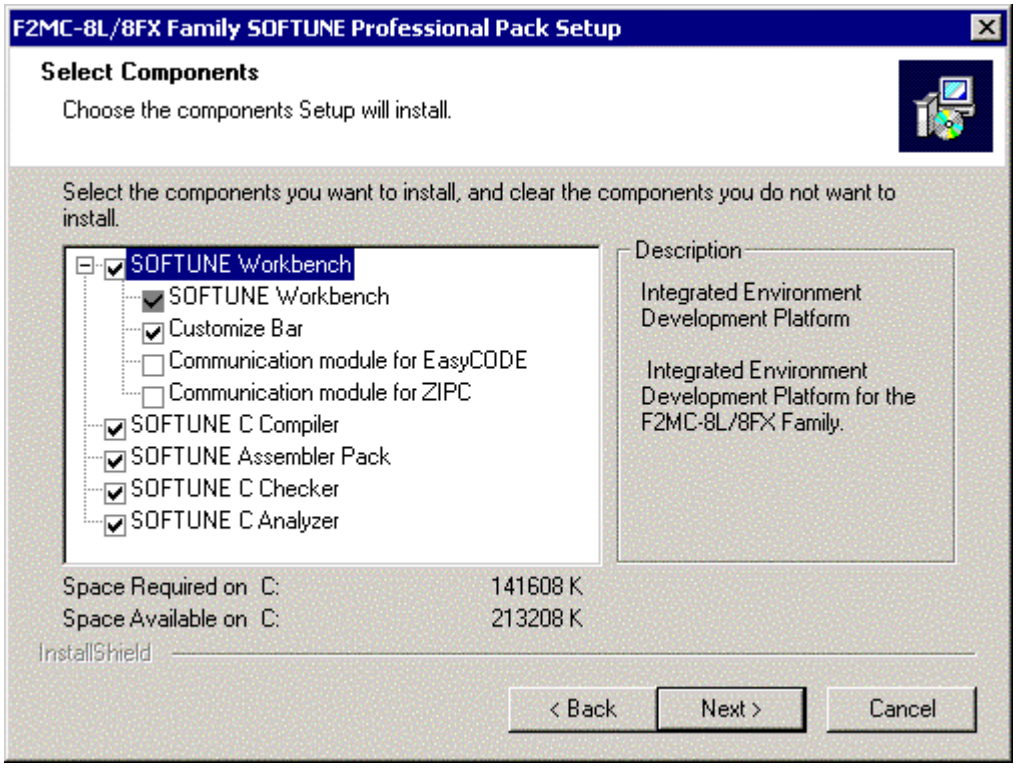
- Refer to the installation manual before installation.
- When installing more than one ***SOFTUNE*** V3 product (for F^2MC-8L, F^2MC-16, and FR Families) on one personal computer, specify the same destination folder.
- When installing ***SOFTUNE*** V3, ***SOFTUNE*** V5 and ***SOFTUNE*** V6 on one personal computer, be sure to specify different destination folders. Installation in the same folder is prohibited.

(4) Select the components to install.

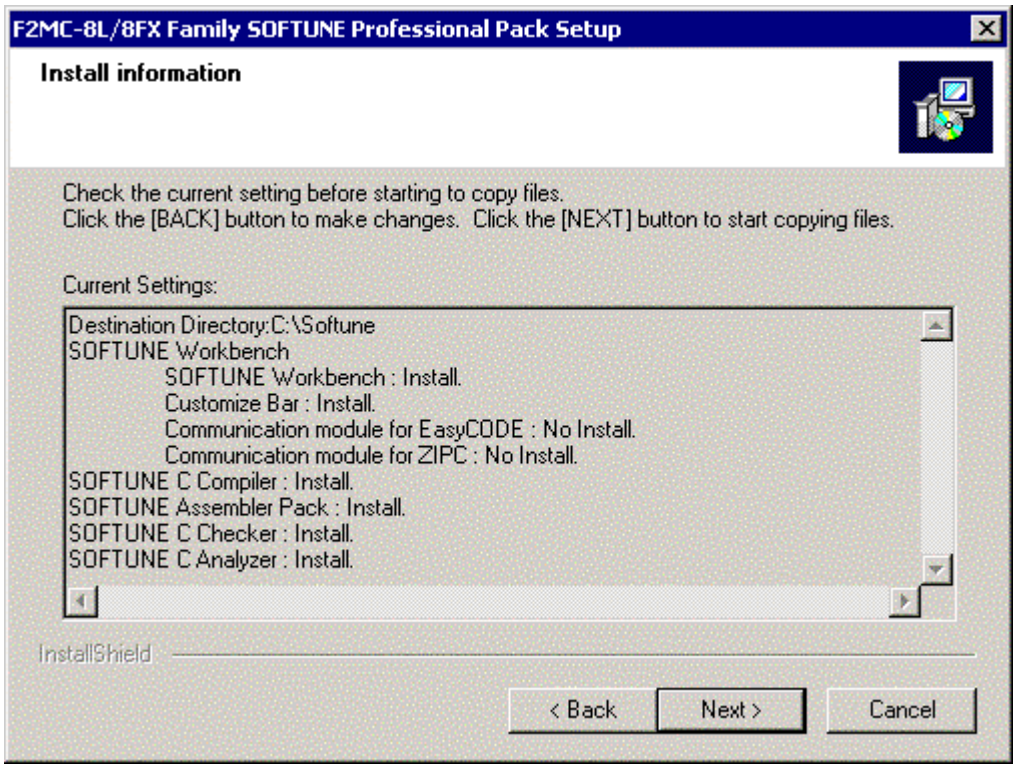
When a component is selected, its description appears at the right side. Mark the appropriate checkboxes.

This manual describes how to operate the following checked tools.

(excluding C Checker and C Analyzer)

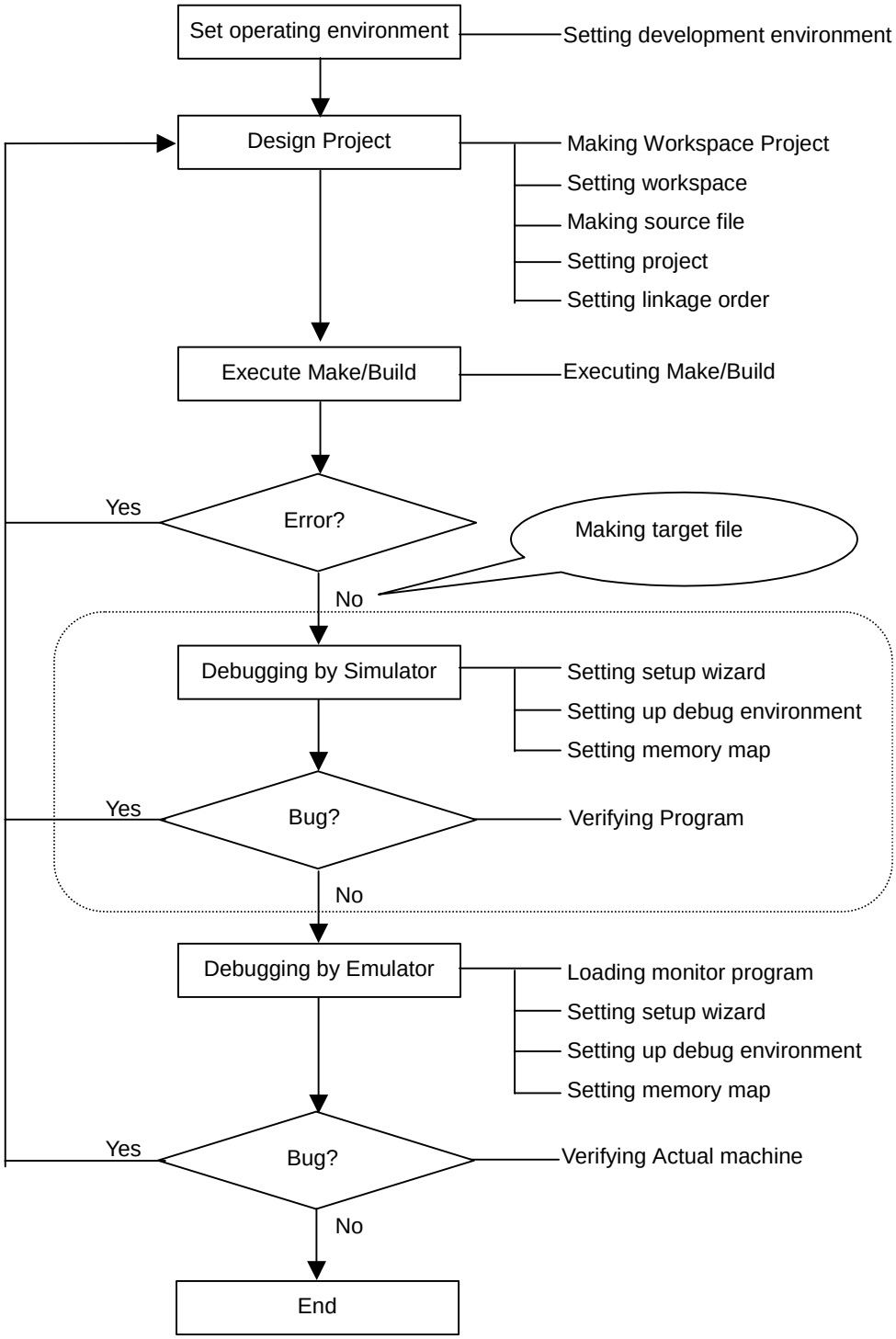


(5) Follow the wizard instruction to check the components to install, and then start installation.



3. DEVELOPMENT PROCEDURE

The software development procedure using ***SOFTUNE*** is given as a flowchart.



Debugging by Simulator may be omitted.

4. TUTORIAL

■ Outline

This chapter gives a concrete example of the software development procedure using ***SOFTUNE*** by using simple samples.
This chapter explains how to make Workspace/Project, debugging by Simulator and Emulator, and how to use the Object Format Converter.

■ 4.1 Designing Workspace/ Project

■ 4.2 Debugging by Simulator

■ 4.3 Debugging by Emulator

■ 4.4 Operate Object Format Converter

4.1 Designing Workspace / Project

This section explains the procedures from making Workspace/Project to making target file creation.
In this sample, the target MCU makes a project of the F^2MC-8FX (MB95FV100).
The explanation assumes that ***SOFTUNE*** Workbench, C Compiler and Assembler pack are installed on C:\ Softune.

- 4.1.1 Setting up Development Environment
- 4.1.2 Makig Workspace and Project
- 4.1.3 Setting Workspace
- 4.1.4 Making Source File
- 4.1.5 Adding Member to Project
- 4.1.6 Setting Project
- 4.1.7 Making Target File

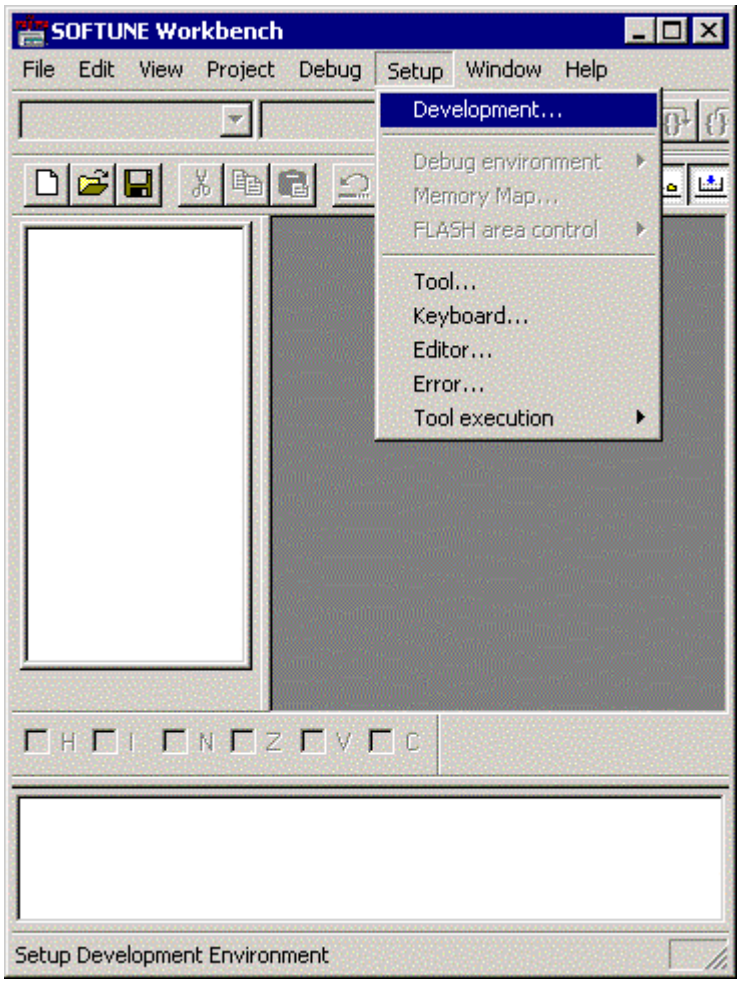
4.1.1 Setting up Development Environment

Set the contents of the [Environment Variable] and [Workspace] tabs in the [Setup Development] dialog.

At installation, the recommended configurations are set in the [Setup Development] dialog.

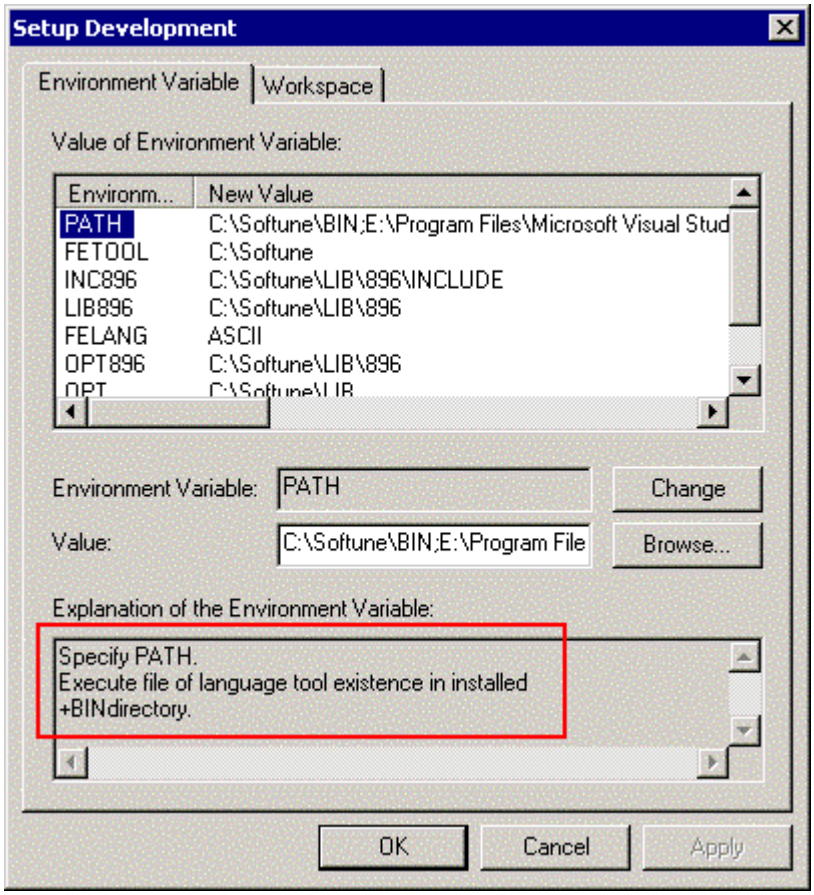
《 Setting procedure 》

(1) Start ***SOFTUNE*** Workbench to select [Development…] in the [Setup] menu.



- **Environment Variable tab**

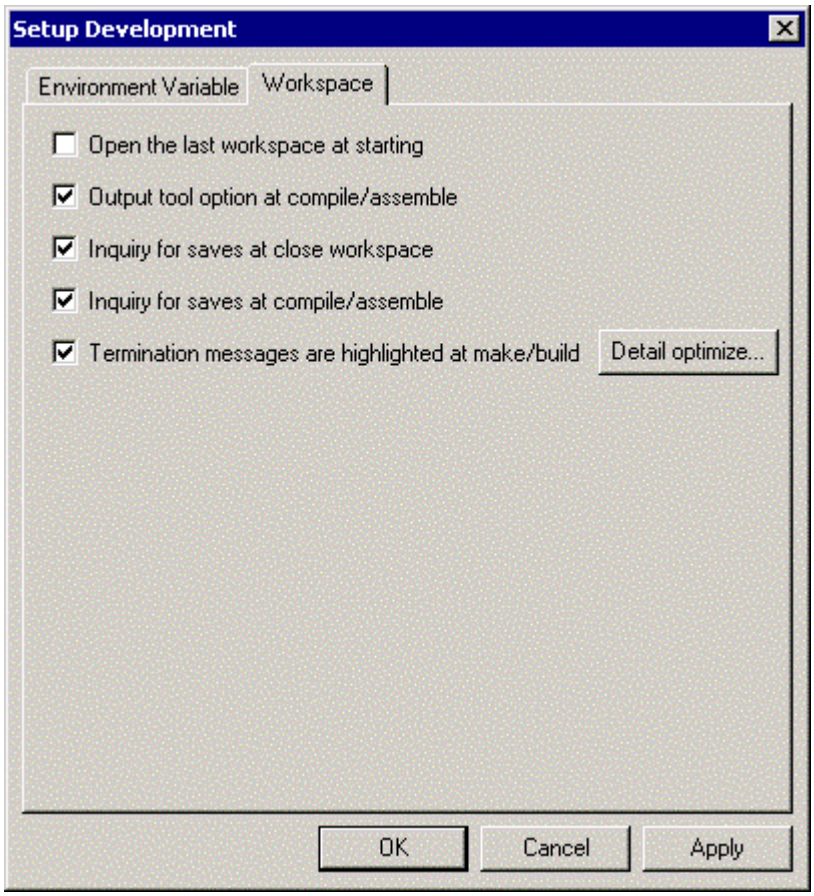
  Set each environment variable. When an environment variable is set, a brief explanation appears.



- **Workspace tab**

  Set the basic conditions required to operate ***SOFTUNE*** Workbench. Mark the appropriate checkboxes.

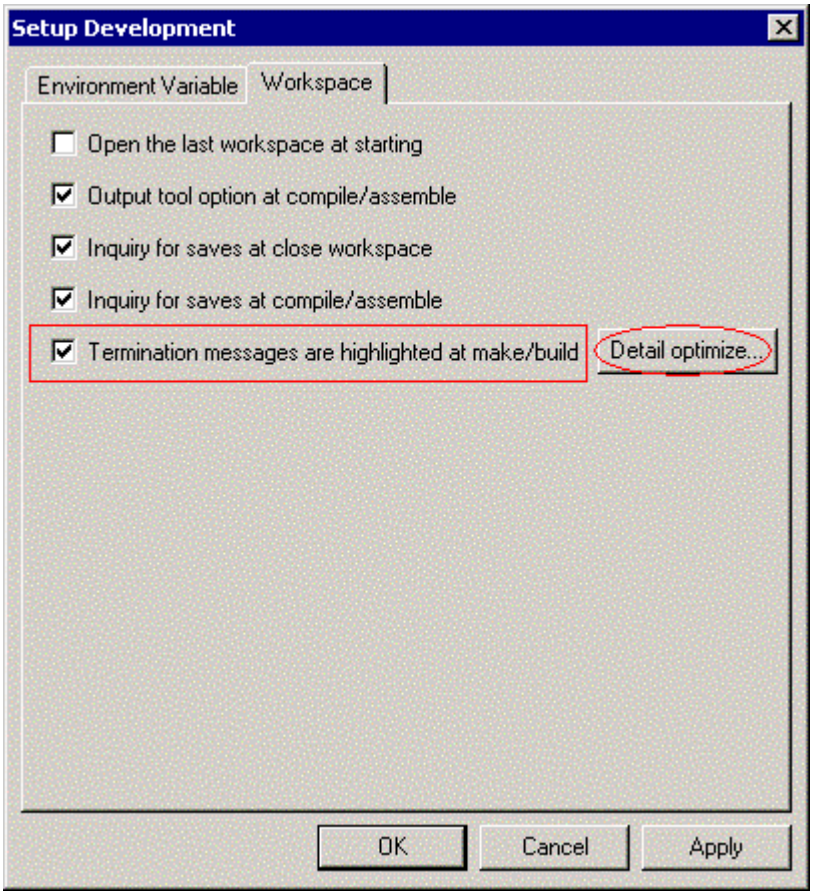
  In this tutorial, mark the checkboxes given in the following dialog.

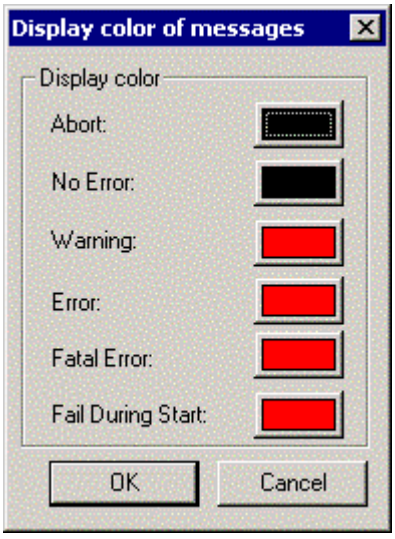


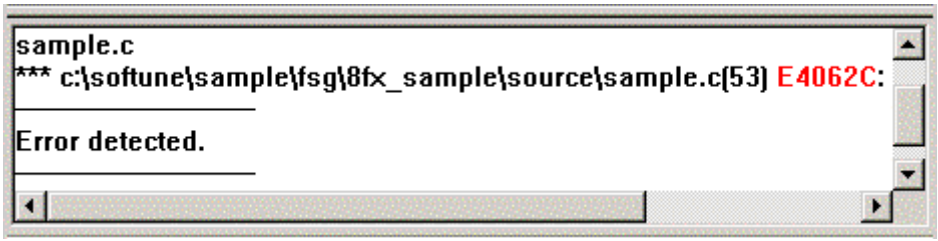
One Point !

= How to change display color of message =

You can change the display color of the termination messages displayed in the Output window at Make/Build execution.
Color-coding makes the massages more visible and readable.
To set the color of termination messages, mark the checkbox for 'Termination messages are highlighted at make/build' in the [Project] tab and click the [Detail optimize...] button.


Setup Development
Environment Variable
Workspace
Open the last workspace at starting
Output tool option at compile/assemble
Inquiry for saves at close workspace
Inquiry for saves at compile/assemble
Termination messages are highlighted at make/build
Detail optimize...
OK
Cancel
Apply


Display color of messages
Display color
Abort:
No Error:
Warning:
Error:
Fatal Error:
Fail During Start:
OK
Cancel


sample.c
*** c:\softune\sample\fsg\8fx_sample\source\sample.c(53) E4062C:
Error detected.

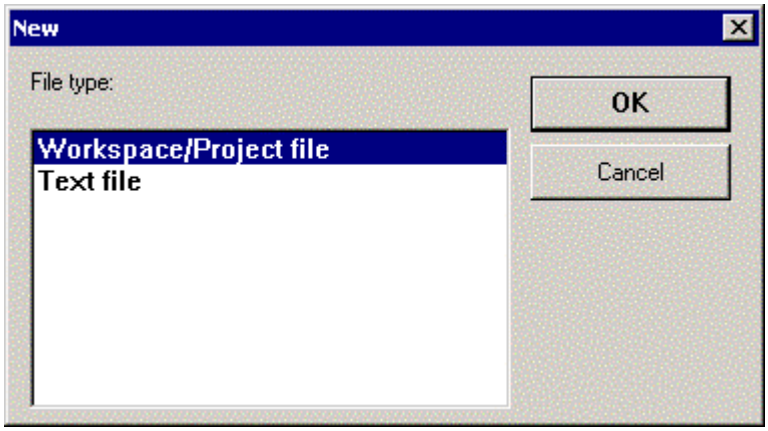
4.1.2 Making Workspace and Project

■ **Making Workspace and Project**

Make workspace and a Project to generate an absolute-type target file.

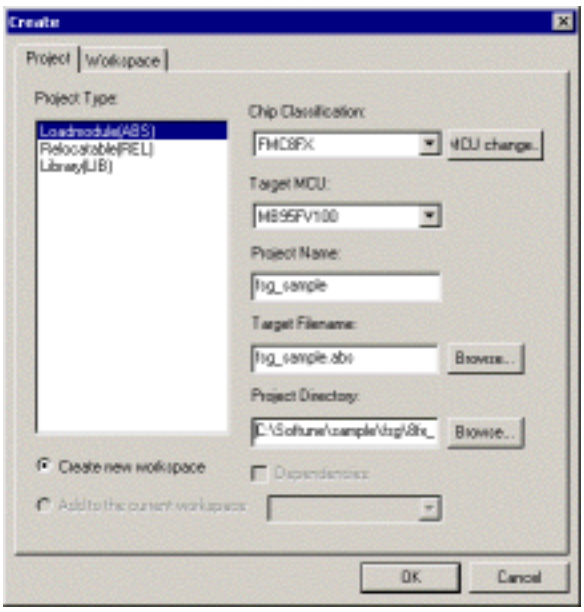
《 Setting procedure 》

(1) Select [Workspace/Project file] from [New] in the [File] menu.

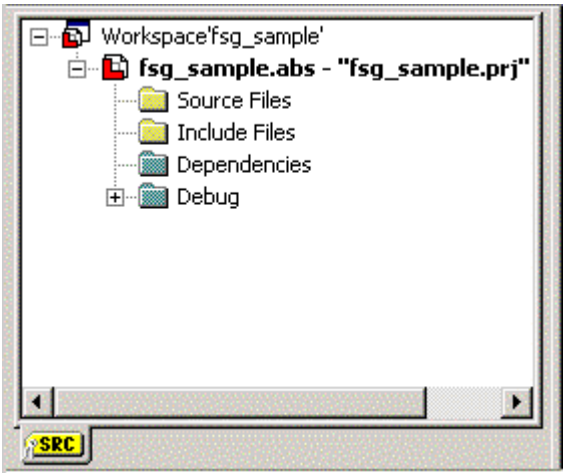


(2) In the [Create] dialog, set the following items and click the [OK] button.

- Project Type → Loadmodule (ABS)
- Chip Classification → FMC8FX
- Target MCU → MB95FV100
- Project Name → fsg_sample
- Target Filename → fsg_sample.abs
- Project Directory → C:\Softune\sample\fsg\8fx_sample



(3) The Project to generate workspace and a target file is made.



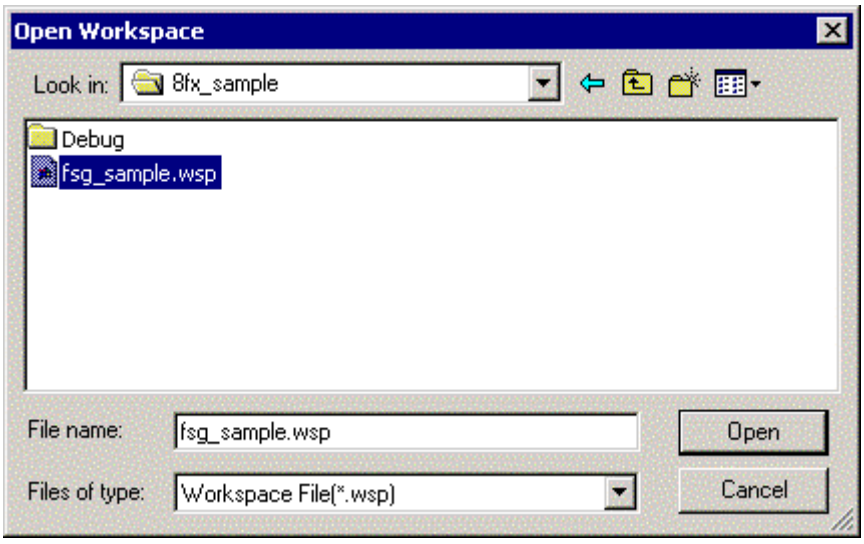
One Point !

= Opening the Workspace/Project =

To open the Project after subsequent Workspace/Project is made, specify the Workspace file. The Workspace file in this sample is **`fsg_sample.wsp`**.

(1) Select [Open Workspace...] from the [File] menu.

(2) Select the Workspace file in the [Open Workspace...] dialog.



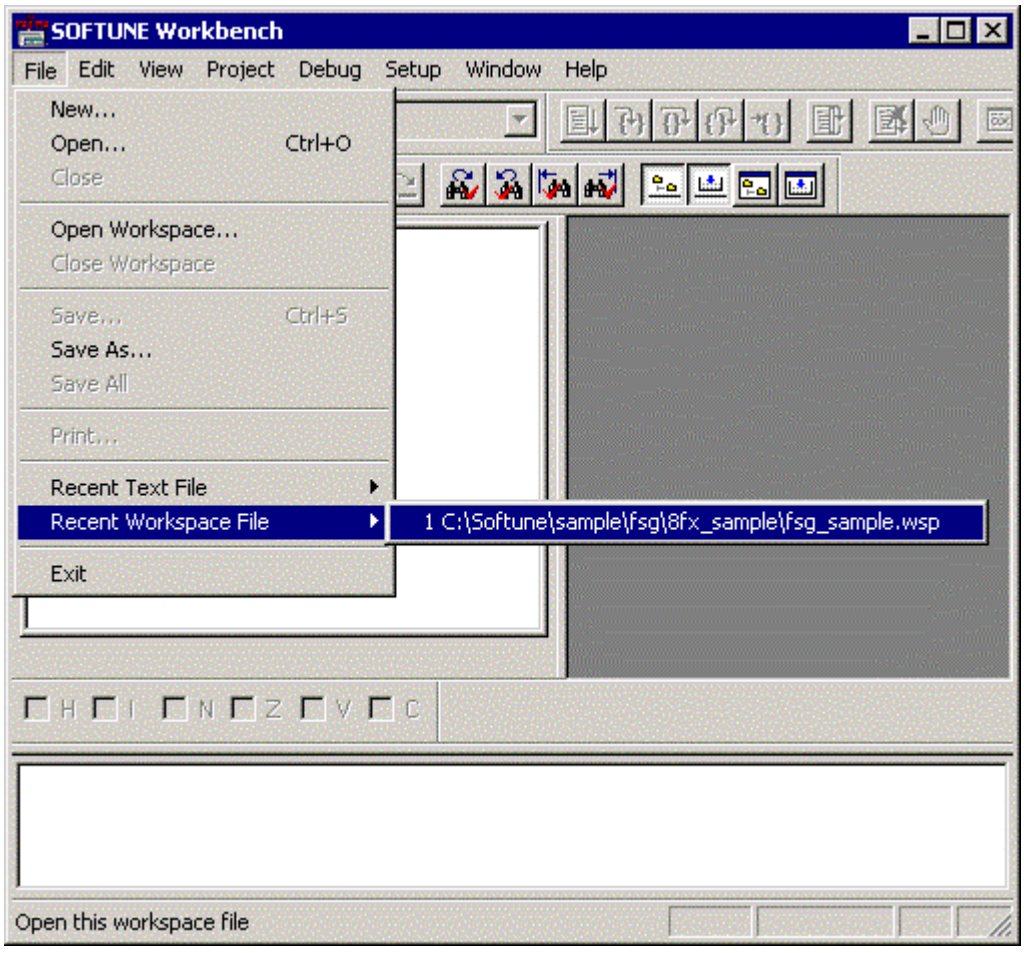
One Point !

= Opening the recently used Workspace file. =

Five Project files are automatically stored, starting with the newest one.

Select [Recent Workspace File] from the [File] menu and choose the workspace file.

Using this button, you need not specify the file name (The same is true for the test file).



4.1.3 Setting Workspace

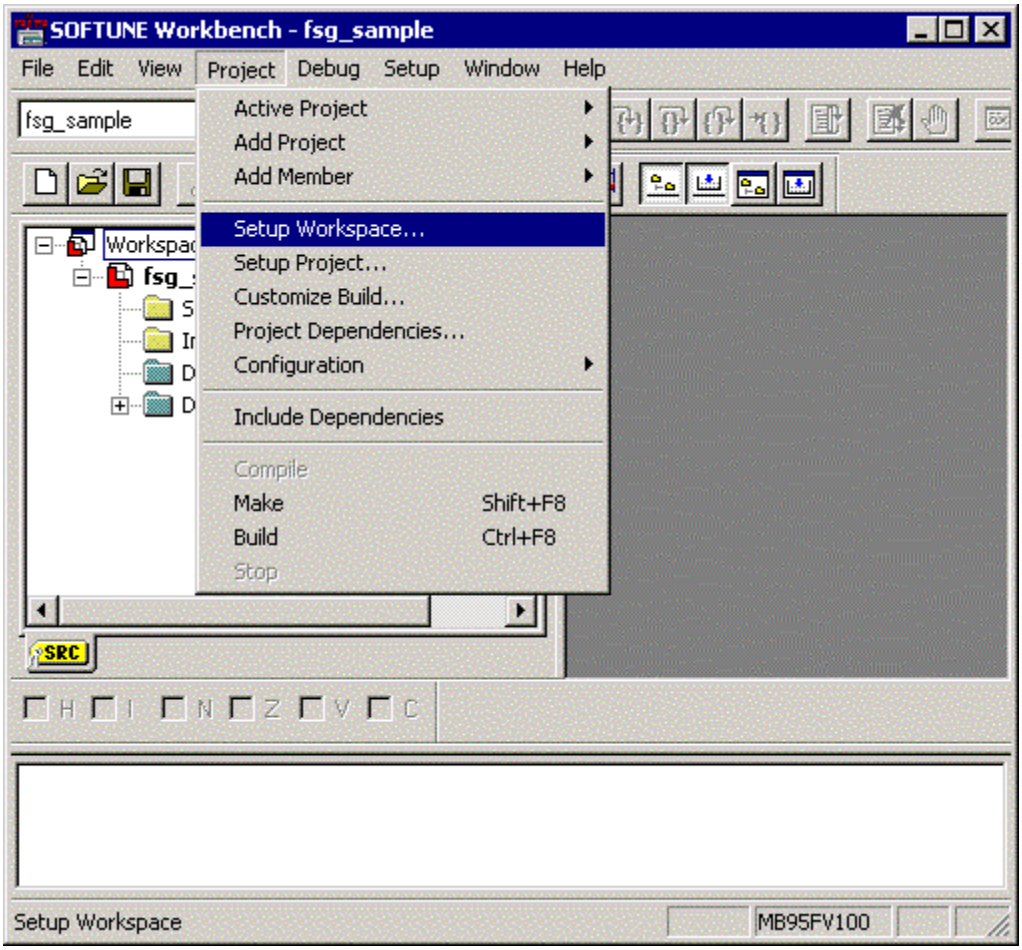
■ Setting Workspace

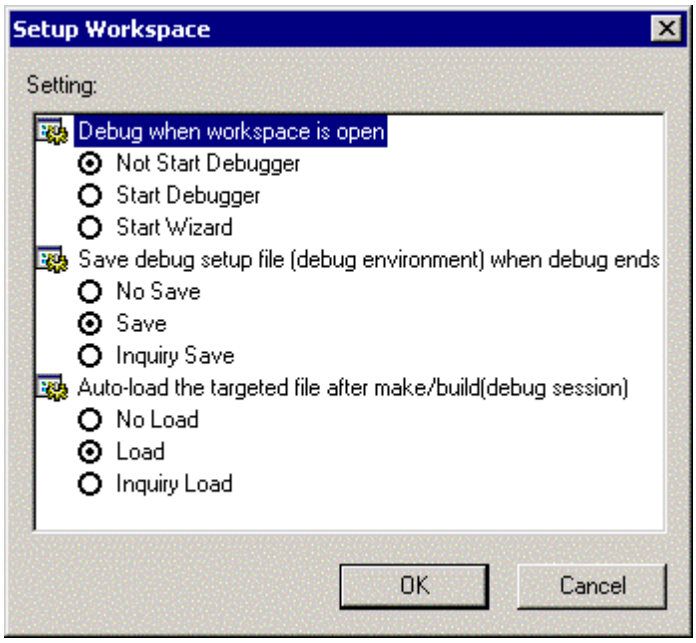
Set items such as start debugging when Workspace is open and save debug environment when debug ends, etc. This sample requires no special setting. The settings are given for reference.

《 Setting procedure 》

(1) Select [Setup workspace...] from the [Project] menu.

Set the necessary items in the [Setup Workspace] dialog.




Setup Workspace
Setting:
Debug when workspace is open
Not Start Debugger
Start Debugger
Start Wizard
Save debug setup file (debug environment) when debug ends
No Save
Save
Inquiry Save
Auto-load the targeted file after make/build(debug session)
No Load
Load
Inquiry Load
OK
Cancel

4.1.4 Making Source File

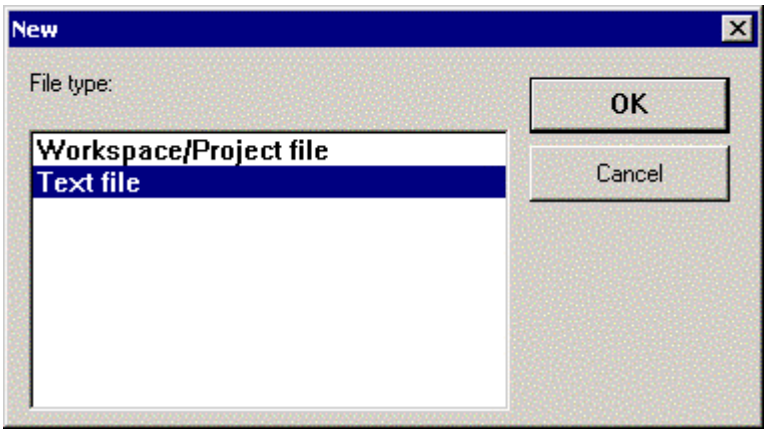
■ Making Source File

This sample uses a simple program that makes LEDs on the target board emit light in regular intervals.

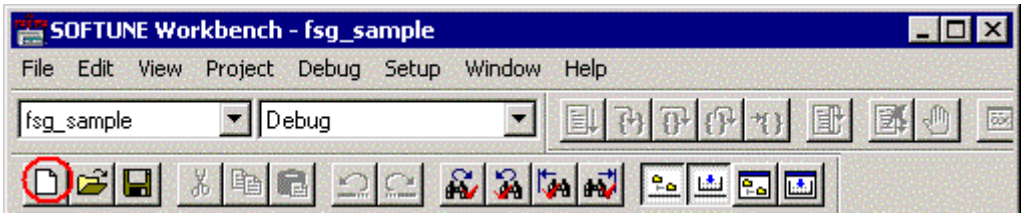
- init.c
- sample.c
- vector.c
- startup.asm
- define.h

《 Operating procedure 》

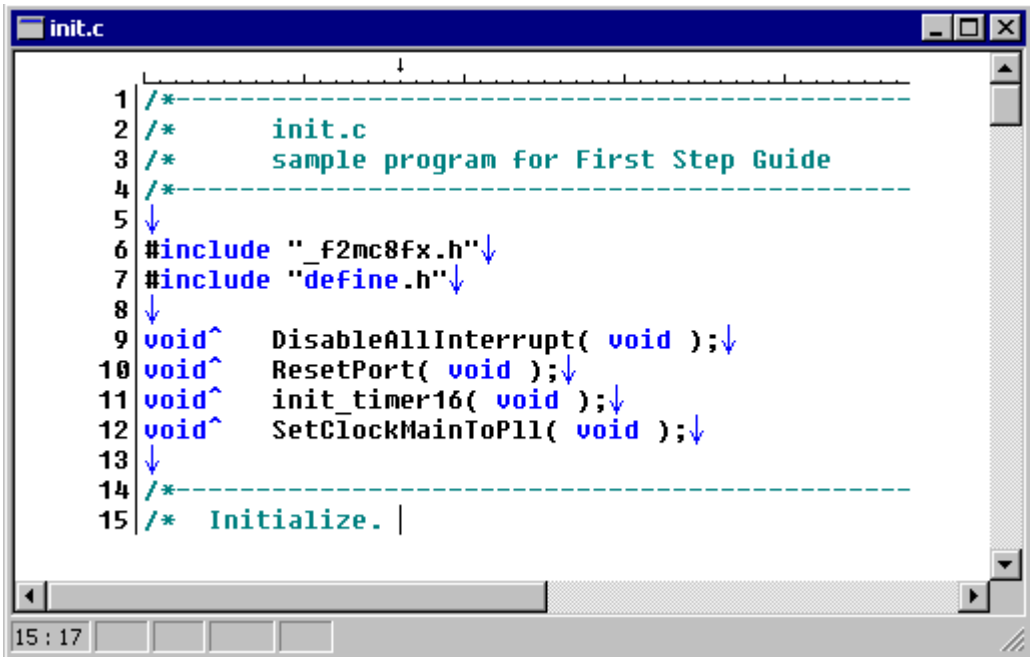
(1) Select [Text file] from [New] in the [File] menu and click the [OK] button.



The following button enclosed in line can also make a new text file.

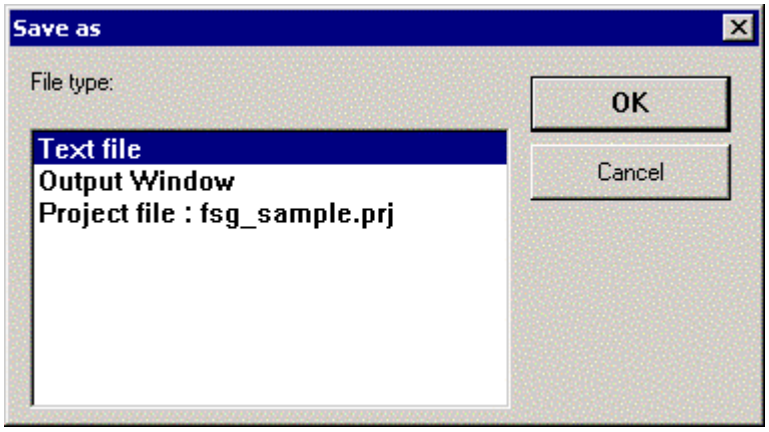


(2) Make a source file.

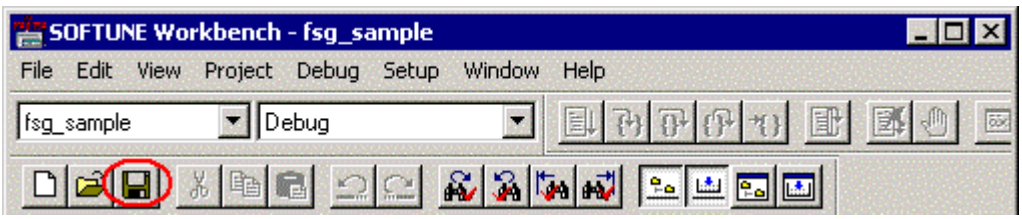


(3) Save the source file.

Select [Text file] from [Save as] in the [File] menu.

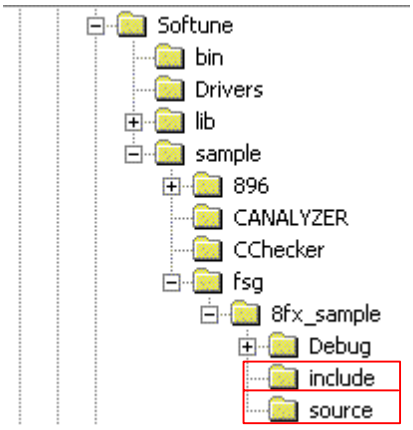


The following button enclosed in line can also save the source file.



(4) Select the file save destination, name the file, and click the [Save] button.

In this sample, create ‘source’ and ‘include’ folders under the Project directory and save the source and include files in the respective folders.



One Point !

= Saving file =

Save the source file under the hierarchy of the Project directory.

For details, see ***APPENDIX A Q1 How do I move the SOFTUNE Workbench project to another machine or drive?***

Sample source init.c

```
/*-----------------------------------------------------------------------*/
/*      init.c                                                           */
/*      sample program for First Step Guide                              */
/*-----------------------------------------------------------------------*/

#include "_f2mc8fx.h"
#include "define.h"

void    DisableAllInterrupt( void );
void    ResetPort( void );
void    init_timer16( void );
void    SetClockMainToPll( void );

/*-----------------------------------------------------------------------*/
/*  Initialize.                                                          */
/*-----------------------------------------------------------------------*/
void Initialize( void )
{
    DisableAllInterrupt();      /* Disable all interrupts */
    ResetPort();                /* Reset the port */
    init_timer16();             /* Initialize 16-bit Reload Timer */
    return;
}

/*-----------------------------------------------------------------------*/
/*  Disable all interrupts.                                              */
/*-----------------------------------------------------------------------*/
void DisableAllInterrupt( void )
{
    __DI();     /* Disable interruption */

    SET_INT_LEVEL_IRQ00(INT_DISABLE);   /* External interrupt (ch0,ch4)   */
    SET_INT_LEVEL_IRQ01(INT_DISABLE);   /* External interrupt (ch1,ch5)   */
    SET_INT_LEVEL_IRQ02(INT_DISABLE);   /* External interrupt (ch2,ch6)   */
    SET_INT_LEVEL_IRQ03(INT_DISABLE);   /* External interrupt (ch3,ch7)   */
    SET_INT_LEVEL_IRQ04(INT_DISABLE);   /* UART/SIO 0                     */
    SET_INT_LEVEL_IRQ05(INT_DISABLE);   /* 8/16 bit Timer 0 L             */
```

```
SET_INT_LEVEL_IRQ06(INT_DISABLE);   /* 8/16 bit Timer 0 H                 */
SET_INT_LEVEL_IRQ07(INT_DISABLE);   /* LIN-UART (REC)                     */
SET_INT_LEVEL_IRQ08(INT_DISABLE);   /* LIN-UART (TRA)                     */
SET_INT_LEVEL_IRQ09(INT_DISABLE);   /* 8/16 bit PPG 1 L, UART/SIO 1       */
SET_INT_LEVEL_IRQ10(INT_DISABLE);   /* 8/16 bit PPG 1 H, I2C 1            */
SET_INT_LEVEL_IRQ11(INT_DISABLE);   /* 16 bit Reload Timer 0, Custom 0    */
SET_INT_LEVEL_IRQ12(INT_DISABLE);   /* 8/16 bit PPG 0 H                   */
SET_INT_LEVEL_IRQ13(INT_DISABLE);   /* 8/16 bit PPG 0 L                   */
SET_INT_LEVEL_IRQ14(INT_DISABLE);   /* 8/16 bit Timer 1 H                 */
SET_INT_LEVEL_IRQ15(INT_DISABLE);   /* 16 bit PPG 0/2                     */
SET_INT_LEVEL_IRQ16(INT_DISABLE);   /* 16 bit Reload Timer 1, I2C 0       */
SET_INT_LEVEL_IRQ17(INT_DISABLE);   /* 16 bit PPG 1                       */
SET_INT_LEVEL_IRQ18(INT_DISABLE);   /* 10 bit A/D                         */
SET_INT_LEVEL_IRQ19(INT_DISABLE);   /* Timebase Timer                     */
SET_INT_LEVEL_IRQ20(INT_DISABLE);   /* Watch Timer/Counter                */
SET_INT_LEVEL_IRQ21(INT_DISABLE);   /* External interrupt (ch8-ch11)      */
SET_INT_LEVEL_IRQ22(INT_DISABLE);   /* 8/16 bit Timer 1 L, Ext interrupt
                                                          (ch12-15)*/
SET_INT_LEVEL_IRQ23(INT_DISABLE);   /* FLASH, Custom 1                    */

/* Disable External interrupt */
IO_EIC00.bit.EIE0 = DISABLE;        /* INT00 */
IO_EIC00.bit.EIE1 = DISABLE;        /* INT01 */
IO_EIC10.bit.EIE0 = DISABLE;        /* INT02 */
IO_EIC10.bit.EIE1 = DISABLE;        /* INT03 */
IO_EIC20.bit.EIE0 = DISABLE;        /* INT04 */
IO_EIC20.bit.EIE1 = DISABLE;        /* INT05 */
IO_EIC30.bit.EIE0 = DISABLE;        /* INT06 */
IO_EIC30.bit.EIE1 = DISABLE;        /* INT07 */
IO_EIC01.bit.EIE0 = DISABLE;        /* INT10 */
IO_EIC01.bit.EIE1 = DISABLE;        /* INT11 */
IO_EIC11.bit.EIE0 = DISABLE;        /* INT12 */
IO_EIC11.bit.EIE1 = DISABLE;        /* INT13 */
IO_EIC21.bit.EIE0 = DISABLE;        /* INT14 */
IO_EIC21.bit.EIE1 = DISABLE;        /* INT15 */
IO_EIC31.bit.EIE0 = DISABLE;        /* INT16 */
IO_EIC31.bit.EIE1 = DISABLE;        /* INT17 */
```

```
    __EI();     /* Enable interrupt */
    return;
}


/*---------------------------------------------------------------------------*/
/* Reset the port.                                                           */
/*---------------------------------------------------------------------------*/
void ResetPort( void )
{
    IO_PDR6.byte = IO_OFF;
    IO_DDR6.byte = IO_OUTPUT;
                            /* P60:     OFF: */
                            /* P61:     OFF: */
                            /* P62:     OFF: */
                            /* P63:     OFF: */
                            /* P64:     OFF: */
                            /* P65:     OFF: */
                            /* P66:     OFF: */
                            /* P67:     OFF: */
    IO_PDR0.byte = IO_OFF;
    IO_DDR0.byte = IO_OUTPUT;
                            /* P00:     OFF: */
                            /* P01:     OFF: */
                            /* P02:     OFF: */
                            /* P03:     OFF: */
                            /* P04:     OFF: */
                            /* P05:     OFF: */
                            /* P06:     OFF: */
                            /* P07:     OFF: */
    return;
}


/*---------------------------------------------------------------------------*/
/*  Initialize 16-bit Reload Timer #0.                                       */
/*---------------------------------------------------------------------------*/
void init_timer16(void)
```

```
{
    __DI();     /* Disable interrupt */

    SET_INT_LEVEL_IRQ11(2);     /* Set the interrupt level.(level = 2)        */

    /* Set the Reload value to 16bits Reload Register */
    IO_TMR0.byte.TMRH = 0xA0;   /* 16bits Reload Register (HIGH)              */
    IO_TMR0.byte.TMRL = 0x00;   /* 16bits Reload Register (LOW)               */

    /* Set Timer Control Register (HIGH) */
    IO_TMCSRH0.bit.MOD0 = 0;    /* Internal clock mode (Disable External Input) */
    IO_TMCSRH0.bit.MOD1 = 0;
    IO_TMCSRH0.bit.MOD2 = 0;
    IO_TMCSRH0.bit.CSL0 = 0;    /* Internal clock (1 / MCLK)                  */
    IO_TMCSRH0.bit.CSL1 = 0;
    IO_TMCSRH0.bit.CSL2 = 0;

    /* Set Timer Control Register (LOW) */
    IO_TMCSRL0.bit.OUTE = 0;    /* Function as general-purpose I/O port       */
    IO_TMCSRL0.bit.RELD = 1;    /* Reload mode                                */
    IO_TMCSRL0.bit.INTE = 1;    /* Permit Under-flow Interruption Request     */
    IO_TMCSRL0.bit.CNTE = 1;    /* Permit the count                           */
    IO_TMCSRL0.bit.TRG  = 1;    /* Start count operation                      */

    __EI();     /* Enable interruption */
    return;
}
/*-------------------------------------------------------------------------*/
/*  Switch the clock from Main Clock to PLL Clock.                         */
/*-------------------------------------------------------------------------*/
void SetClockMainToPll(void)
{
    /* Set PLL Control Register */
    /* (Main Clock Oscillation Frequency [FCH = 4MHz]) */
    IO_PLLC.bit.MPMC0 = CLOCK_X2_0; /* Set the PLL multiplication rate.  */
    IO_PLLC.bit.MPMC1 = CLOCK_X2_1; /*              [4MHz=(FCH/2)*2]     */
    IO_PLLC.bit.MPEN  = ENABLE;     /* Enable Main PLL Clock Oscillation */
    /* Wait until Main PLL Clock Oscillation is stabilized  */
```

```
    while(IO_PLLC.bit.MPRDY == 0) {;}

    /* Set System Clock Control Register */
    IO_SYCC.bit.DIV0 = CLOCK_DIV1_0;    /* Select System Clock Dividing Ratio.*/
    IO_SYCC.bit.DIV1 = CLOCK_DIV1_1;    /*              [No Dividing]         */
    IO_SYCC.bit.SCS0 = MAIN_PLL_CLOCK_MODE_0;   /* Select Main PLL Clock Mode */
    IO_SYCC.bit.SCS1 = MAIN_PLL_CLOCK_MODE_1;
    while( !((IO_SYCC.bit.SCM0 == MAIN_PLL_CLOCK_MODE_0) &&
             (IO_SYCC.bit.SCM1 == MAIN_PLL_CLOCK_MODE_1)) ) {;}

    return;
}
```

Sample source sample.c

```
/*---------------------------------------------------------------------------*/
/*      sample.c                                                             */
/*      sample program for First Step Guide                                  */
/*---------------------------------------------------------------------------*/

#include "_f2mc8fx.h"
#include "define.h"

#define LEFT  1
#define RIGHT 2

unsigned short flag = 0x0001;
unsigned char counter = 0x00;
unsigned char direction = LEFT;

extern void Initialize(void);

const unsigned char  LED_pat[9] = {
                     0xff, /* -------- */
                     0xfe, /* -------* */
                     0xfd, /* ------*- */
                     0xfb, /* -----*-- */
                     0xf7, /* ----*--- */
                     0xef, /* ---*---- */
                     0xdf, /* --*----- */
                     0xbf, /* -*------ */
                     0x7f, /* *------- */
              };

void main(void)
{

       Initialize(); /* Initialize the system */
       while(1) {
       }
}
```

```
/*---------------------------------------------------------------------------*/
/* Interrupt handling for 16-bit Reload Timer 0                              */
/*---------------------------------------------------------------------------*/

__interrupt void timer_int(void)
{
       IO_TMCSRL0.bit.UF = 0;             /* Clear the request flag */
       if (++counter > 5) {
              counter = 0;
              /* Determine value to be output to LEDs according to flag value */
              switch (flag) {
              case 0x0001:
                     IO_PDR0.byte = LED_pat[1];
                     break;
              case 0x0002:
                     IO_PDR0.byte = LED_pat[2];
                     break;
              case 0x0004:
                     IO_PDR0.byte = LED_pat[3];
                     break;
              case 0x0008:
                     IO_PDR0.byte = LED_pat[4];
                     break;
              case 0x0010:
                     IO_PDR0.byte = LED_pat[5];
                     break;
              case 0x0020:
                     IO_PDR0.byte = LED_pat[6];
                     break;
              case 0x0040:
                     IO_PDR0.byte = LED_pat[7];
                     break;
              case 0x0080:
                     IO_PDR0.byte = LED_pat[8];
                     IO_PDR6.byte = LED_pat[0];
                     break;
              case 0x0100:
```

```
        IO_PDR0.byte = LED_pat[0];
        IO_PDR6.byte = LED_pat[1];
        break;
case 0x0200:
        IO_PDR6.byte = LED_pat[2];
        break;
case 0x0400:
        IO_PDR6.byte = LED_pat[3];
        break;
case 0x0800:
        IO_PDR6.byte = LED_pat[4];
        break;
case 0x1000:
        IO_PDR6.byte = LED_pat[5];
        break;
case 0x2000:
        IO_PDR6.byte = LED_pat[6];
        break;
case 0x4000:
        IO_PDR6.byte = LED_pat[7];
        break;
case 0x8000:
        IO_PDR6.byte = LED_pat[8];
        break;
}
switch (direction) {
case LEFT:
        if (!(flag <<= 1)) { /* Move LED output digit to left */
                flag = 0x8000;
                direction = RIGHT;
        }
        break;
case RIGHT:
        if (!(flag >>= 1)) { /* Move LED output digit to right */
                flag = 0x0001;
                direction = LEFT;
        }
        break;
```

```
                }
        }
}
```

Sample source vector.c

```
/*-----------------------------------------------------------------------*/
/*      vector.c                                                         */
/*      sample program for First Step Guide                              */
/*-----------------------------------------------------------------------*/
/*-----------------------------------------------------------------------*/
/* Set interrupt function in interrupt vector.                           */
/*-----------------------------------------------------------------------*/
               /* Reference declaration of the interrupt function to be set */
extern __interrupt void timer_int(void);
                     /* Add interrupt vector (11) of 16-bit reload timer 0 */
#pragma intvect timer_int 11

/*-----------------------------------------------------------------------*/
/* Set the startup function in reset vector.                             */
/*-----------------------------------------------------------------------*/
                /* Reference declaration of the startup function to be set. */
extern void _start(void);
                /* Set RESETVECT section attribute to CONST   */
#pragma section CONST=RESETVECT, locate=0xfffc
void (* const reset_vector[2])() = {
              (void (*)())(0x00),
              _start
};
```

Sample source startup.asm

```
;===============================================================================
; F2MC-8FX Family SOFTUNE First Step Guide startup routine,
; ALL RIGHTS RESERVED, COPYRIGHT (C) FUJITSU LIMITED 2004
; LICENSED MATERIAL - PROGRAM PROPERTY OF FUJITSU LIMITED
;===============================================================================
;  Sample program for initialization
;-------------------------------------------------------------------------------
              .PROGRAM      start
              .TITLE        start

;-------------------------------------------------------------------------------
; external declaration of symbols
;-------------------------------------------------------------------------------
              .EXPORT       __start
              .IMPORT       _main
              .IMPORT       _SetClockMainToPll
              .IMPORT       LMEMTOMEM
              .IMPORT       LMEMCLEAR
              .IMPORT       _RAM_INIT
              .IMPORT       _ROM_INIT
              .IMPORT       _RAM_DIRINIT
              .IMPORT       _ROM_DIRINIT

;-------------------------------------------------------------------------------
; definition to stack area
;-------------------------------------------------------------------------------
              .SECTION      STACK,    STACK,    ALIGN=1
              .RES.B        254
STACK_TOP:
              .RES.B        2

;-------------------------------------------------------------------------------
; definition to start address of data, const and code section
;-------------------------------------------------------------------------------
              .SECTION      DIRDATA,  DIR,      ALIGN=1
DIRDATA_S:
              .SECTION      DIRINIT,  DIR,      ALIGN=1
```

```
DIRINIT_S:
              .SECTION      DATA,     DATA,     ALIGN=2
DATA_S:
              .SECTION      INIT,     DATA,     ALIGN=2
INIT_S:

;------------------------------------------------------------------------
; code area
;------------------------------------------------------------------------
              .SECTION      CODE,     CODE,     ALIGN=1
__start:
;------------------------------------------------------------------------
; set stack pointer
;------------------------------------------------------------------------
              MOVW   A, #STACK_TOP
              MOVW   SP, A
;------------------------------------------------------------------------
; set register bank is 0
;------------------------------------------------------------------------
              MOVW   A, PS
              MOVW   A, #0x07FF
              ANDW   A
              MOVW   PS, A
;------------------------------------------------------------------------
; set direct bank pointer
;------------------------------------------------------------------------
;             A default setup is B'000. (Direct address: 0x0080..0x00FF)

;------------------------------------------------------------------------
; set I flag
;------------------------------------------------------------------------
;             A default setup is B'0. (Interruption disable)

;------------------------------------------------------------------------
; set ILM to the lowest level(3)
;------------------------------------------------------------------------
              MOVW   A, PS
              MOVW   A, #0x0030
```

```
                ORW     A
                MOVW    PS, A
;-------------------------------------------------------------------------
; call SetClockMainToPll routine
;-------------------------------------------------------------------------
                CALL    _SetClockMainToPll
;-------------------------------------------------------------------------
; copy initial value *CONST(ROM) section to *INIT(RAM) section
;-------------------------------------------------------------------------
#macro          ICOPY   src_addr, dest_addr, src_segment
                MOVW    EP, #\src_addr
                MOVW    A,  #\dest_addr
                MOVW    A,  #SIZEOF (\src_segment)
                CALL    LMEMTOMEM
#endm

                ICOPY   _ROM_INIT,     _RAM_INIT, INIT
                ICOPY   _ROM_DIRINIT, _RAM_DIRINIT, DIRINIT

;-------------------------------------------------------------------------
; zero clear of *VAR section
;-------------------------------------------------------------------------
#macro          FILL0   src_addr, src_segment
                MOVW    A, #\src_addr
                MOVW    A, #SIZEOF (\src_segment)
                CALL    LMEMCLEAR
#endm

                FILL0   DIRDATA_S, DIRDATA
                FILL0   DATA_S,    DATA

;-------------------------------------------------------------------------
; call main routine
;-------------------------------------------------------------------------
                CALL    _main
end:            JMP     end

                .END    __start
```

Sample source define.h

```
/*---------------------------------------------------------------------------*/
/* define.h                                                                  */
/* sample program for First Step Guide                                       */
/*---------------------------------------------------------------------------*/
#ifndef __define_h__
#define __define_h__

#define DISABLE         0x00
#define ENABLE          0x01

/*---------------------------------------------------------------------------*/
/* Definition for I/O port data register                                     */
/*---------------------------------------------------------------------------*/
#define IO_ON           0x00
#define IO_OFF          0xFF

/*---------------------------------------------------------------------------*/
/* Definition for I/O port direction register                                */
/*---------------------------------------------------------------------------*/
#define IO_OUTPUT       0xFF
#define IO_INPUTPUT     0x00

/*---------------------------------------------------------------------------*/
/* Definition for interrupt level setting register                           */
/*---------------------------------------------------------------------------*/
#define INT_DISABLE     0x03

/*---------------------------------------------------------------------------*/
/* Macro for interrupt level setting register                                */
/*---------------------------------------------------------------------------*/
#define SET_INT_LEVEL_IRQ00(level) (IO_ILR0 &= ((((level) & 0x03))  | 0xFC))
#define SET_INT_LEVEL_IRQ01(level) (IO_ILR0 &= ((((level) & 0x03) << 2) | 0xF3))
#define SET_INT_LEVEL_IRQ02(level) (IO_ILR0 &= ((((level) & 0x03) << 4) | 0xCF))
#define SET_INT_LEVEL_IRQ03(level) (IO_ILR0 &= ((((level) & 0x03) << 6) | 0x3F))
#define SET_INT_LEVEL_IRQ04(level) (IO_ILR1 &= ((((level) & 0x03))  | 0xFC))
#define SET_INT_LEVEL_IRQ05(level) (IO_ILR1 &= ((((level) & 0x03) << 2) | 0xF3))
#define SET_INT_LEVEL_IRQ06(level) (IO_ILR1 &= ((((level) & 0x03) << 4) | 0xCF))
```

```
#define SET_INT_LEVEL_IRQ07(level) (IO_ILR1 &= ((((level) & 0x03) << 6) | 0x3F))
#define SET_INT_LEVEL_IRQ08(level) (IO_ILR2 &= ((((level) & 0x03))  | 0xFC))
#define SET_INT_LEVEL_IRQ09(level) (IO_ILR2 &= ((((level) & 0x03) << 2) | 0xF3))
#define SET_INT_LEVEL_IRQ10(level) (IO_ILR2 &= ((((level) & 0x03) << 4) | 0xCF))
#define SET_INT_LEVEL_IRQ11(level) (IO_ILR2 &= ((((level) & 0x03) << 6) | 0x3F))
#define SET_INT_LEVEL_IRQ12(level) (IO_ILR3 &= ((((level) & 0x03))  | 0xFC))
#define SET_INT_LEVEL_IRQ13(level) (IO_ILR3 &= ((((level) & 0x03) << 2) | 0xF3))
#define SET_INT_LEVEL_IRQ14(level) (IO_ILR3 &= ((((level) & 0x03) << 4) | 0xCF))
#define SET_INT_LEVEL_IRQ15(level) (IO_ILR3 &= ((((level) & 0x03) << 6) | 0x3F))
#define SET_INT_LEVEL_IRQ16(level) (IO_ILR4 &= ((((level) & 0x03))  | 0xFC))
#define SET_INT_LEVEL_IRQ17(level) (IO_ILR4 &= ((((level) & 0x03) << 2) | 0xF3))
#define SET_INT_LEVEL_IRQ18(level) (IO_ILR4 &= ((((level) & 0x03) << 4) | 0xCF))
#define SET_INT_LEVEL_IRQ19(level) (IO_ILR4 &= ((((level) & 0x03) << 6) | 0x3F))
#define SET_INT_LEVEL_IRQ20(level) (IO_ILR5 &= ((((level) & 0x03))  | 0xFC))
#define SET_INT_LEVEL_IRQ21(level) (IO_ILR5 &= ((((level) & 0x03) << 2)  | 0xF3))
#define SET_INT_LEVEL_IRQ22(level) (IO_ILR5 &= ((((level) & 0x03) << 4)  | 0xCF))
#define SET_INT_LEVEL_IRQ23(level) (IO_ILR5 &= ((((level) & 0x03) << 6)  | 0x3F))

/*-----------------------------------------------------------------------------*/
/* Definition for system clock control register                                */
/*-----------------------------------------------------------------------------*/
#define SUB_CLOCK_MODE_0           0
#define SUB_CLOCK_MODE_1           0
#define SUB_PLL_CLOCK_MODE_0       1
#define SUB_PLL_CLOCK_MODE_1       0
#define MAIN_CLOCK_MODE_0          0
#define MAIN_CLOCK_MODE_1          1
#define MAIN_PLL_CLOCK_MODE_0      1
#define MAIN_PLL_CLOCK_MODE_1      1

#define CLOCK_DIV1_0        0                  /* Source Clock / 1   */
#define CLOCK_DIV1_1        0
#define CLOCK_DIV8_0        1                  /* Source Clock / 4   */
#define CLOCK_DIV8_1        0
#define CLOCK_DIV16_0       0                  /* Source Clock / 8   */
#define CLOCK_DIV16_1       1
#define CLOCK_DIV32_0       1                  /* Source Clock / 16  */
#define CLOCK_DIV32_1       1
```

```
/*--------------------------------------------------------------------------*/
/* Definition for PLL control register                                      */
/*--------------------------------------------------------------------------*/
#define CLOCK_X1_0          0      /* Original oscillation clock x 1     */
#define CLOCK_X1_1          0
#define CLOCK_X2_0          1      /* Original oscillation clock x 2     */
#define CLOCK_X2_1          0
#define CLOCK_X2p5_0        0      /* Original oscillation clock x 2.5   */
#define CLOCK_X2p5_1        1

#endif /* __define_h__ */
```

4.1.5 Adding Member to Project

■ **Adding a Member**

Add the made source file to the project Member.

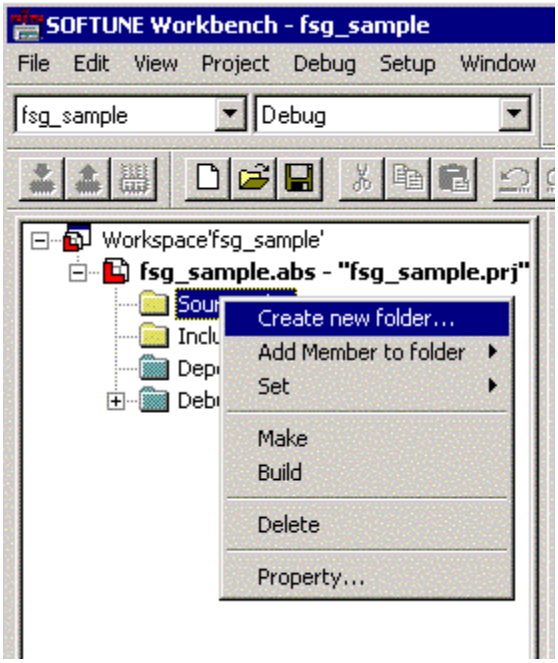
Make a project that generates an I/O area module (**`io_def.rel`**) as a subproject depending on the **`fsg_sample`** project.

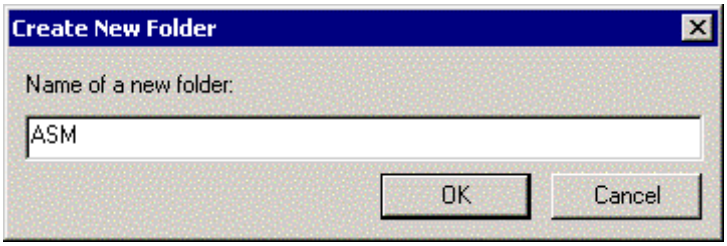
《 Operating procedure 》

(1) Select the **`Source Files`** folder and select [Create New Folder...] from the shortcut menu.

Create the folders "**`ASM`**" and "**`C`**" in the [Create New Folder] dialog.

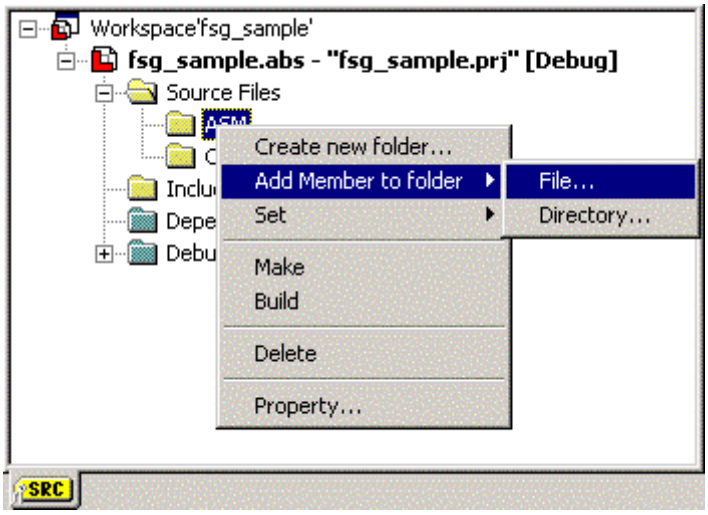
Because, these folders are used to help the users manage files in the ***SOFTUNE*** project, the users can name the folders optionally.

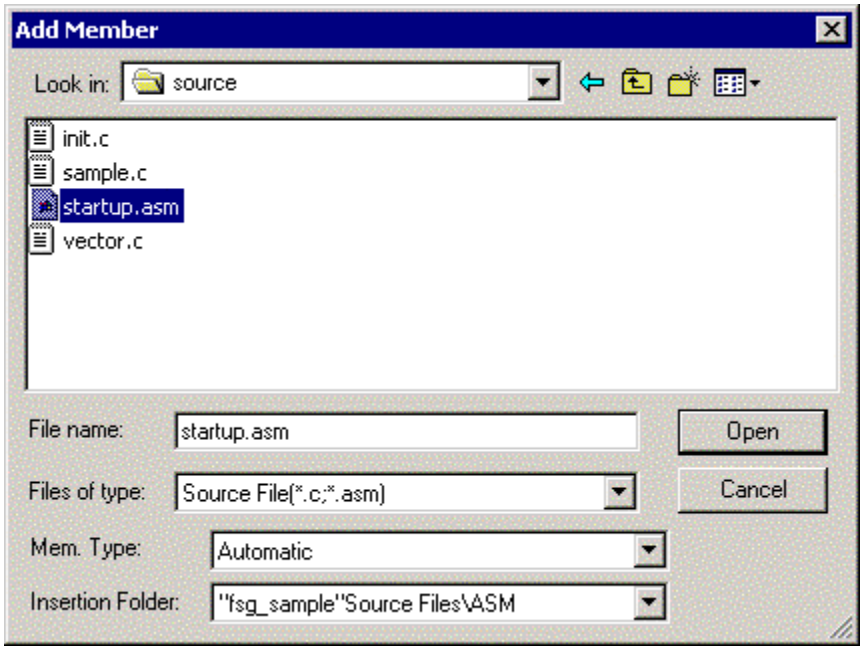




(2) Select [Add Member to folder]-[File] from the shortcut menu for the "**ASM**" and "**C**" folders.

Add the file to the Member in the [Add Member] dialog.

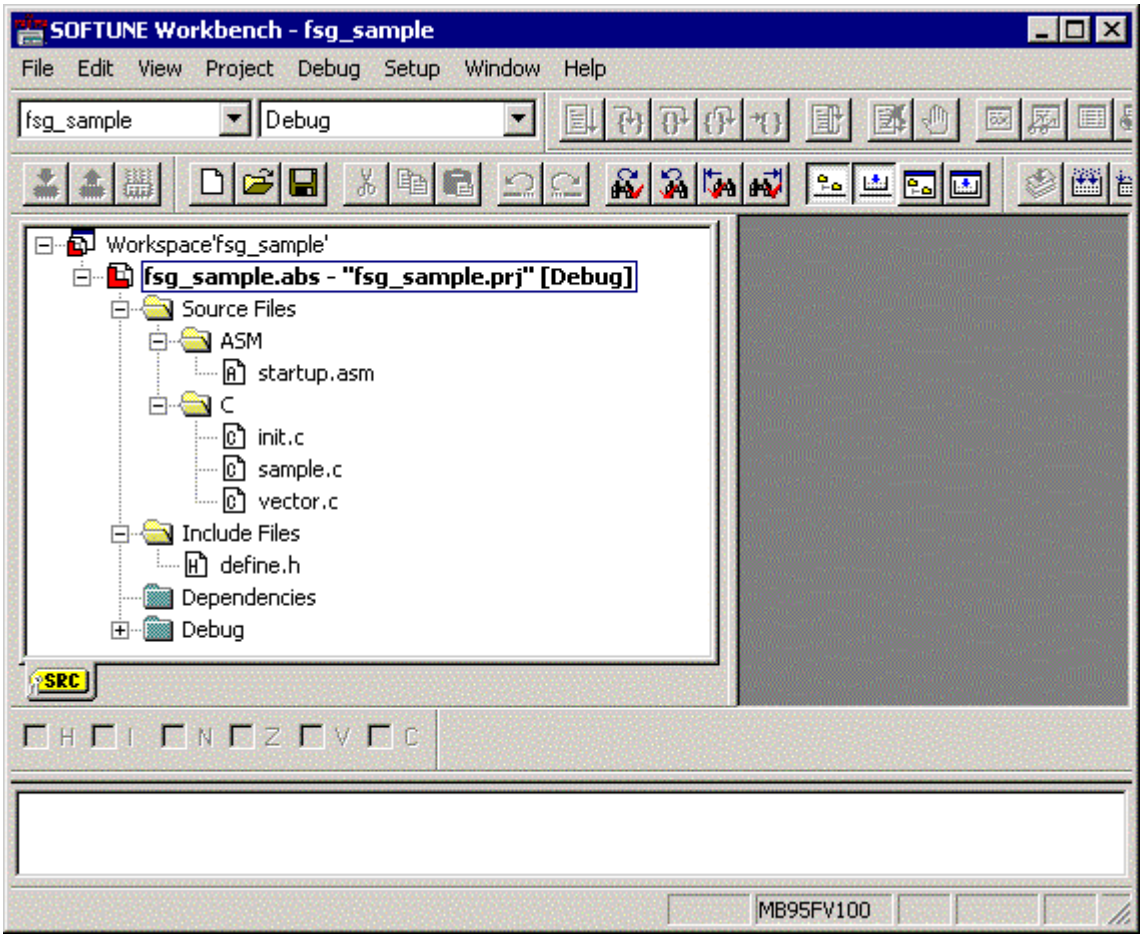




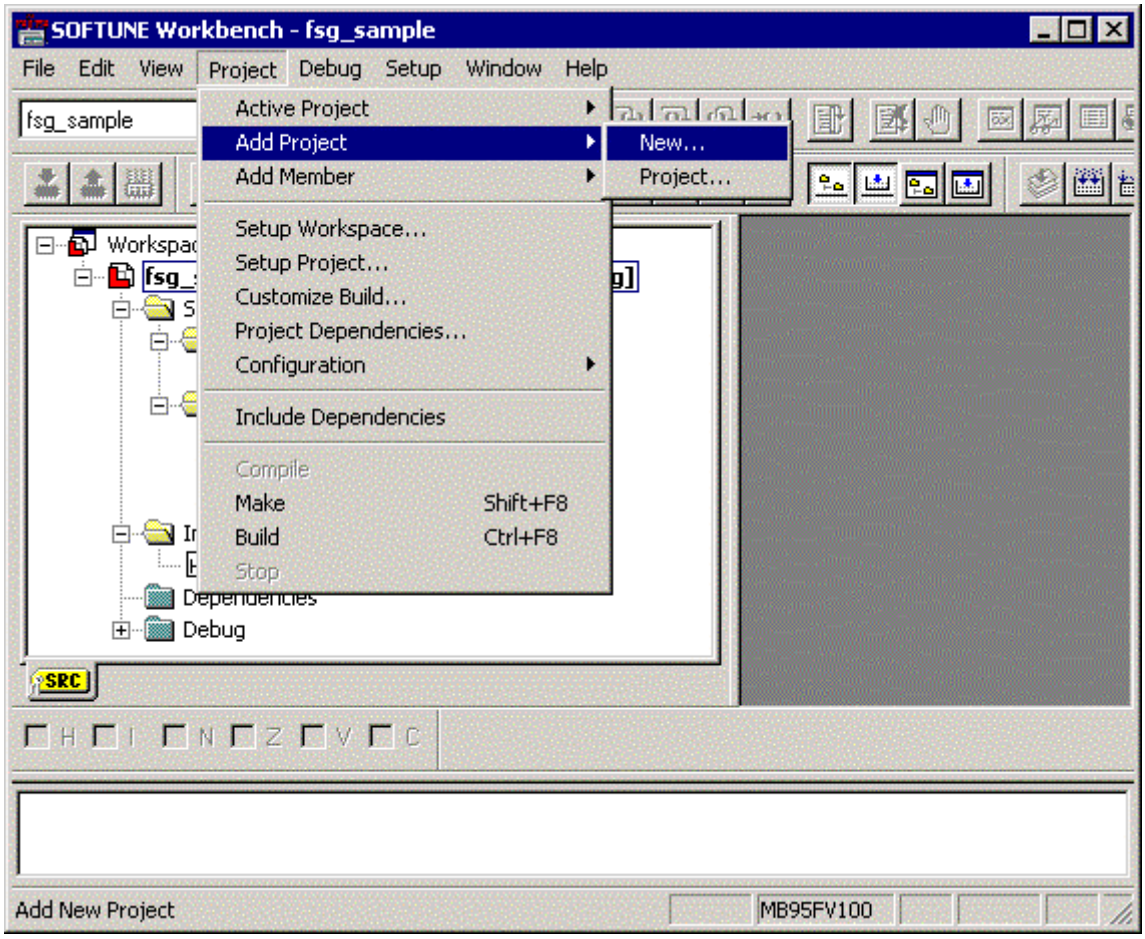
(3) An include file can be added to the Member.

Add the include file to the **`Include Files`** folder in the same way as a member.

The include file is read automatically from the directory set using the include path of the language tool at the Make or Build operation. Therefore, the file does not always have to be added to the Member. (The file is added to the **`Dependencies`** folder when automatically read.)

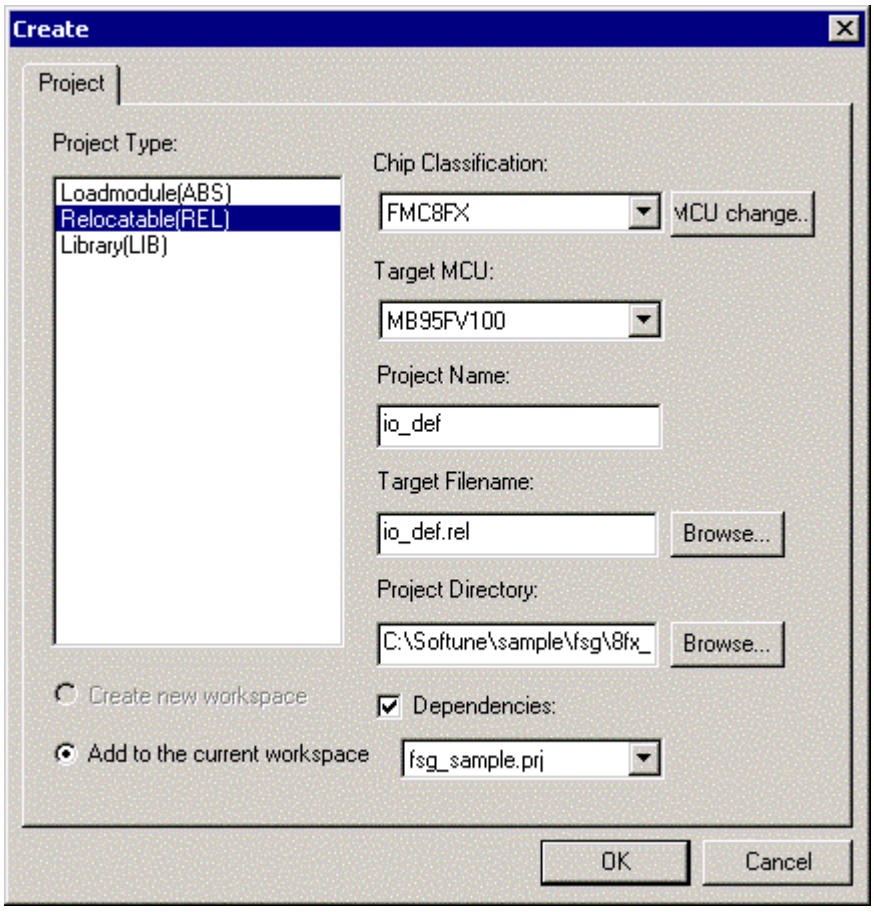


(4) Select [Add Project]-[New] from the [Project] menu.



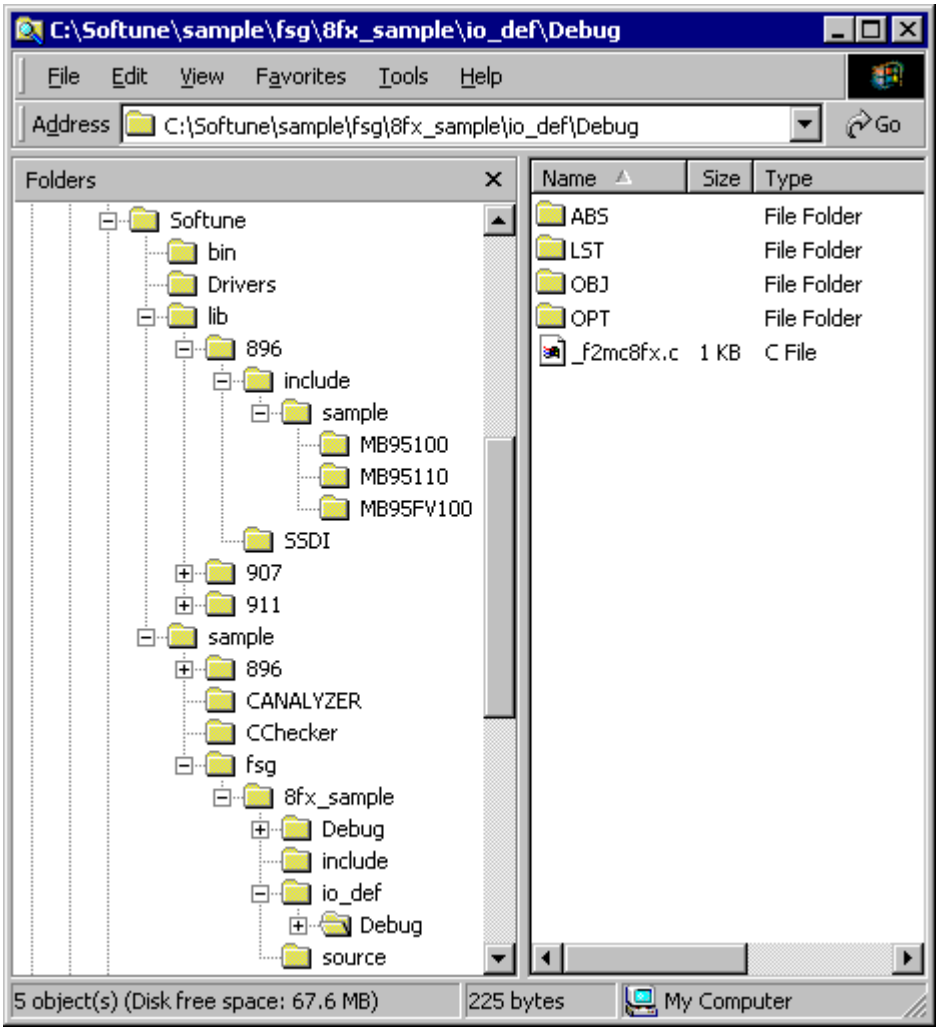
(5) Set the following items in the [Create] dialog.

- Project Type → Relocatable (REL)
- Chip Type → FMC8FX
- Target MCU → MB95FV100
- Project Name → io_def
- Target Filename → io_def.rel
- Project Directory → C:\Softune\sample\fsg\8fx_sample\io_def
- Dependencies → Checkbox ON



(6) Copy a sample I/O register files to the directory below the project directory of the I/O area module creation project.

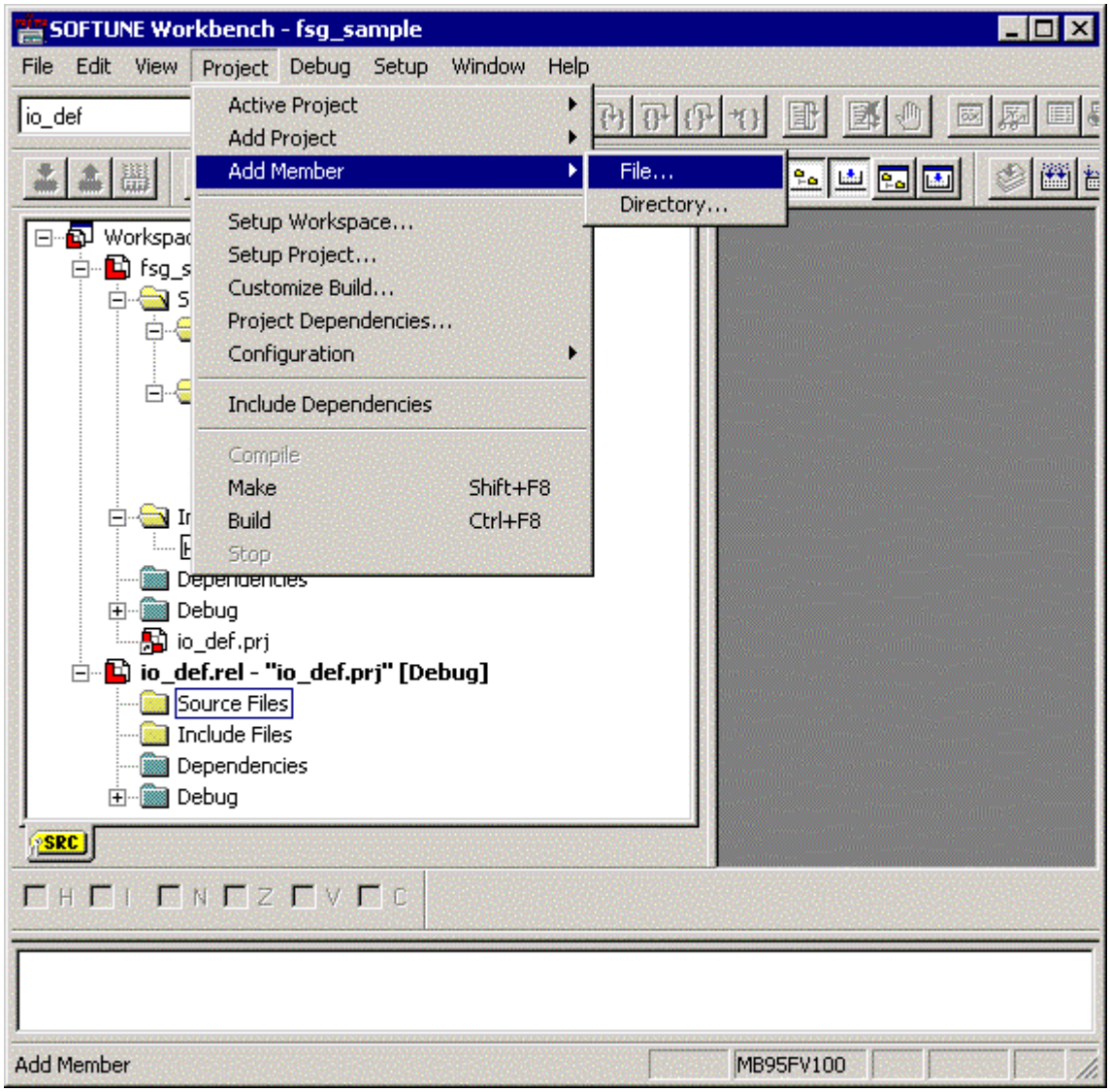
Copy all C source files (.c) in C: \Softune\lib\896\include\sample\MB95FV100 to C:\Softune\sample\fsg\8fx_sample\io_def\Debug.

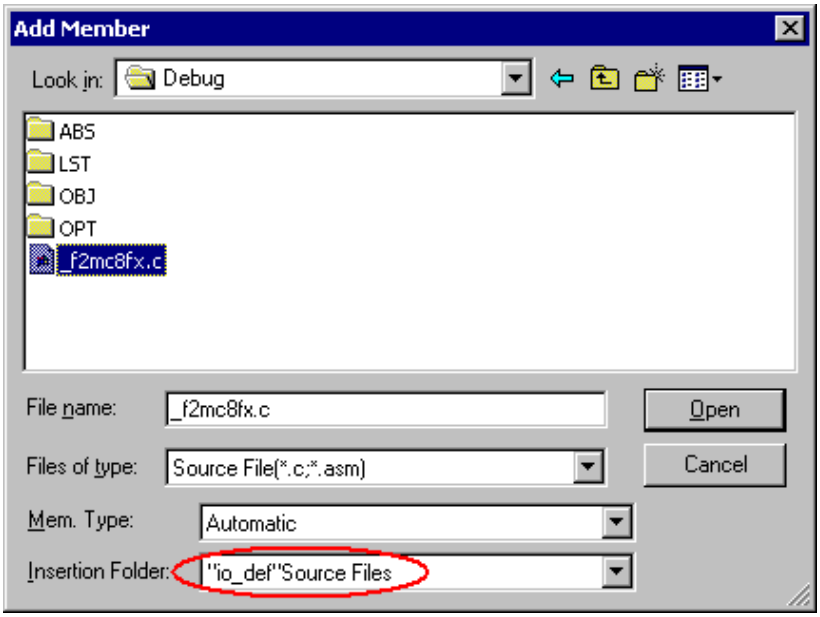


(7) Set a member required to an I/O area.

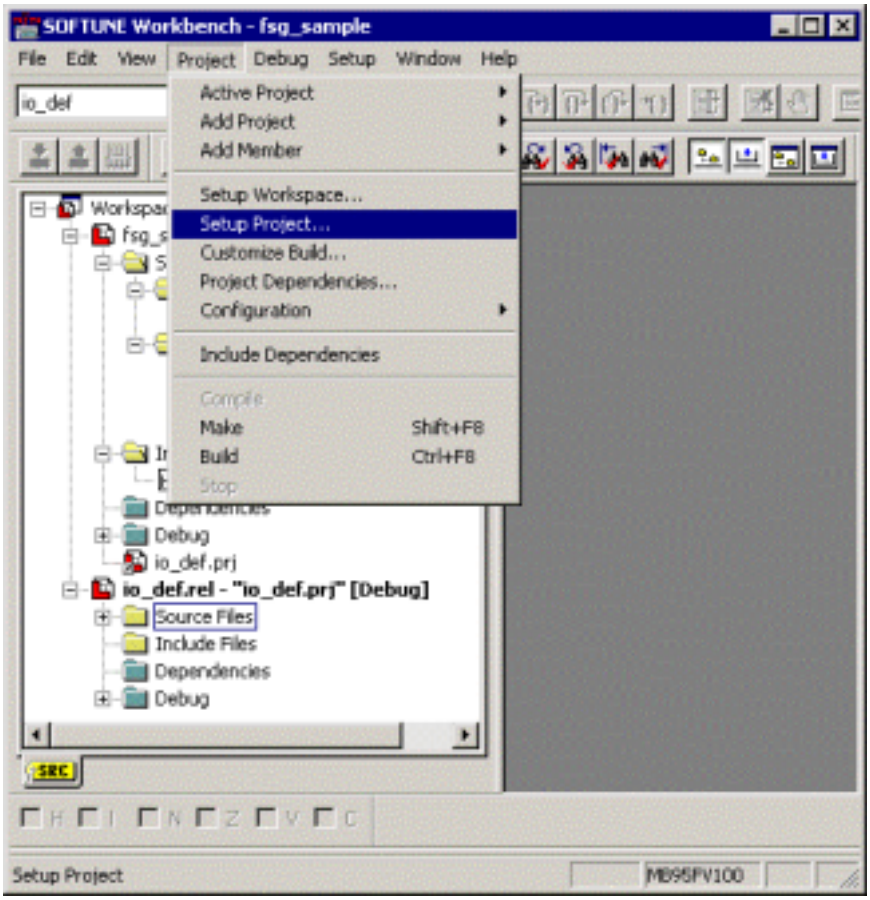
Select [Add Member]-[File] from the [Project] menu.

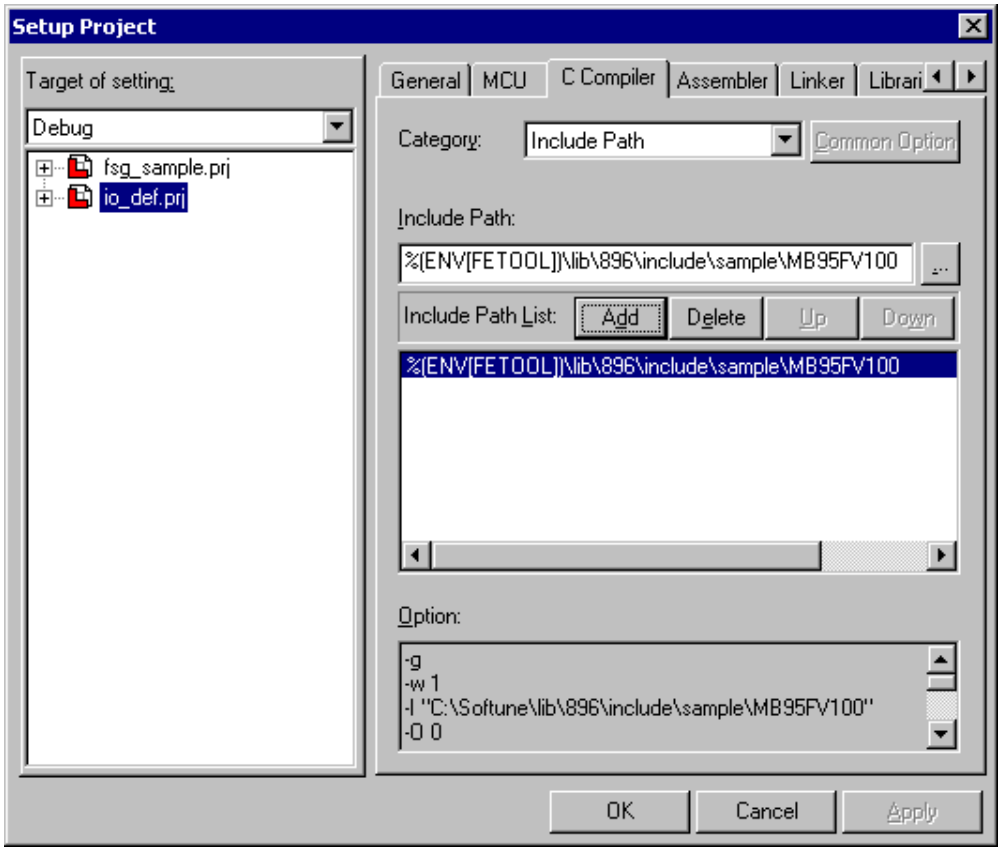
Select all the sample I/O register files copied in step (6) and click the [Open] button. The Member is then added to the project. Specify **`io_def Source Files`** for insertion.



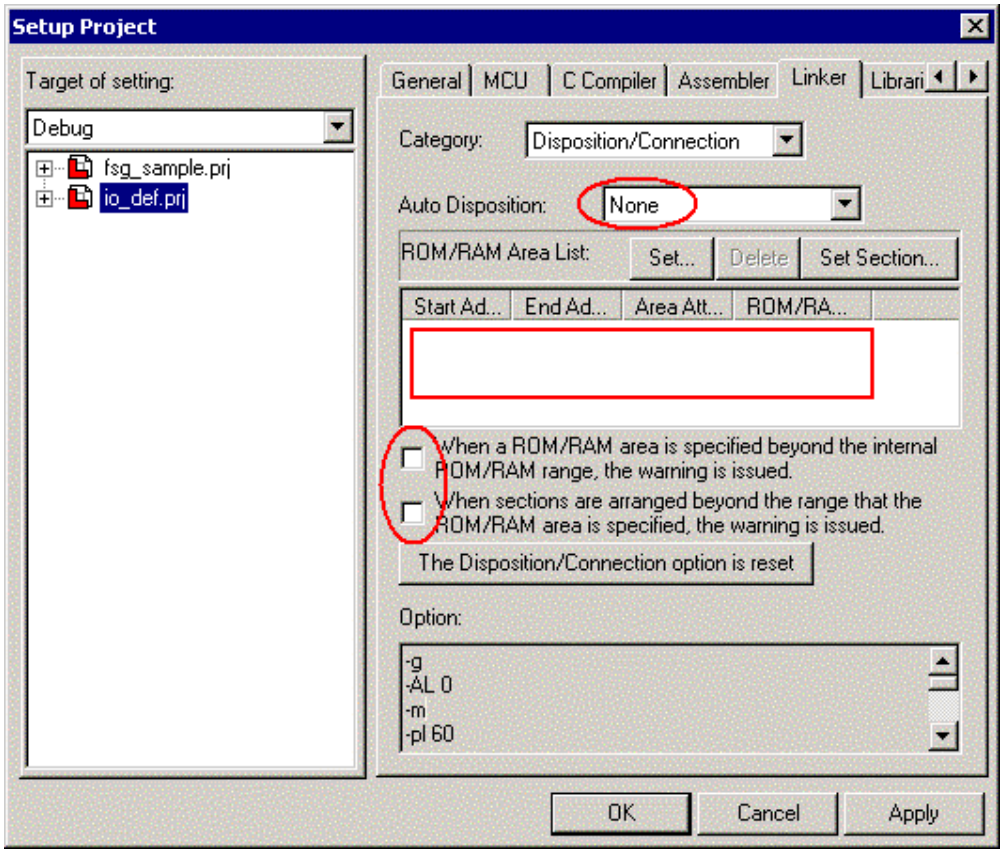


(8) Select [Setup Project] from the [Project] menu.
Click the C Compiler tab in the [Setup Project] dialog and select the Category [Include Path].
Specify the include path using the following macro and click the [Add] button to add the path to the [Include Path List]. For details about the macro, see Section ***4.16 Setting Project***.
%(ENV[FETooL])\lib\896\include\sample\MB95FV100

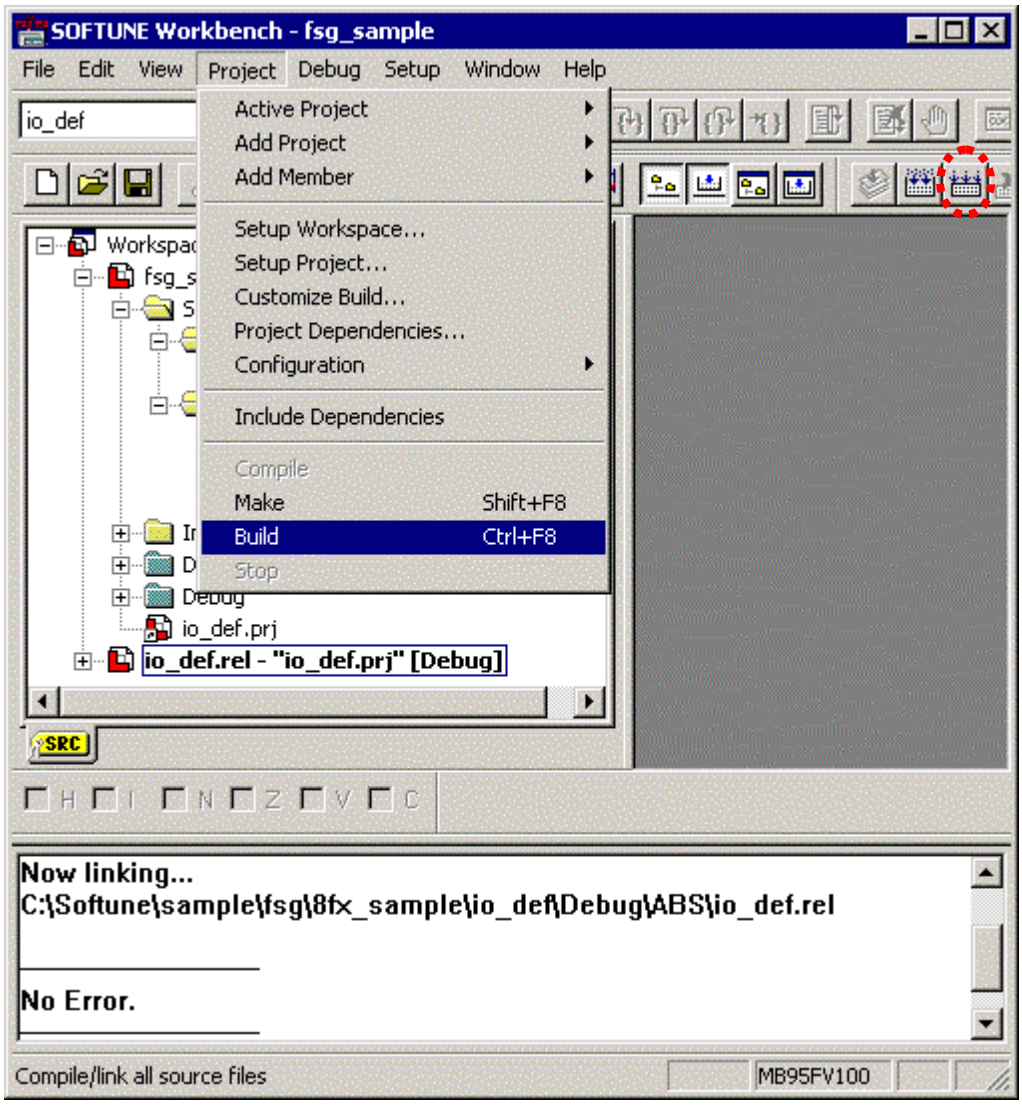




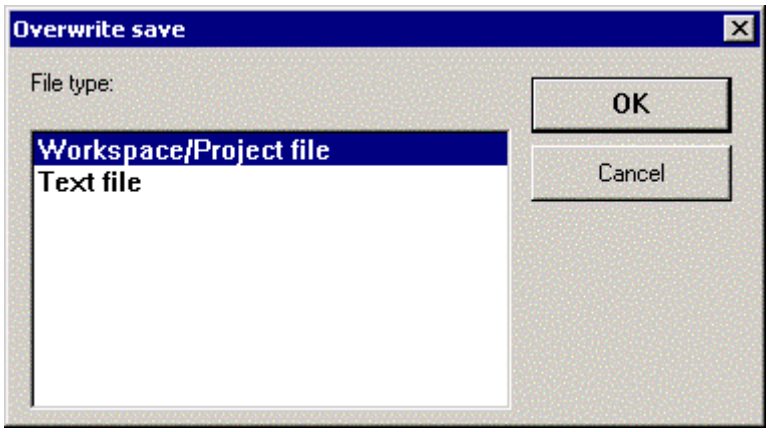
(9) Select the Category [Disposition/Connection] in the Linker tab and disable all options.



(10) Execute [Build] in the [Project] menu (clicking the icon enclosed in the dotted lines also enables [Build]). When the Build operation is terminated without error, **`io_def.rel`** is generated in the C:\Softune\Sample\fsg\8fx_sample\io_def\Debug\ABS directory.



(11) Save the Project.
Select [Save]-[Workspace/Project File] from the [File] menu and click the [OK] button.



One Point !

= Caution for customizing I/O register file =

Reinstallation of ***SOFTUNE*** Workbench overwrites the sample I/O register file. ALWAYS copy the file to another directory before customization.

4.1.6 Setting Project

■ **Procedure for Changing Project Settings**

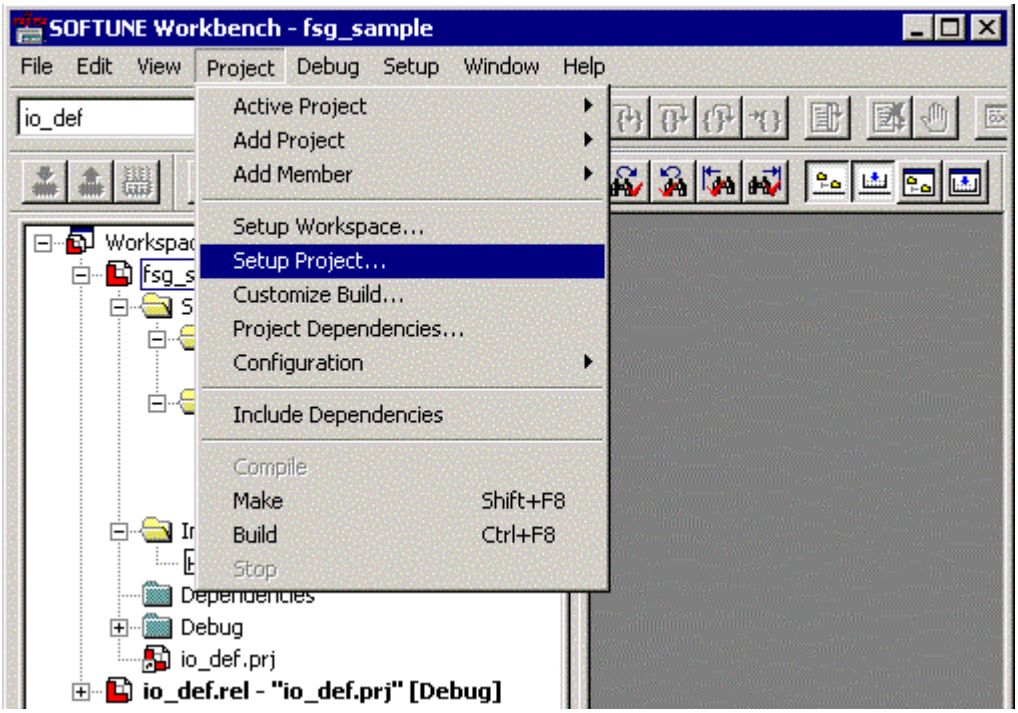
Select the Project to be set and set the MCU and various language tool options, etc. If necessary, add or change the settings for making a Project and the default tool option settings.

The following setting method is explained for this sample.

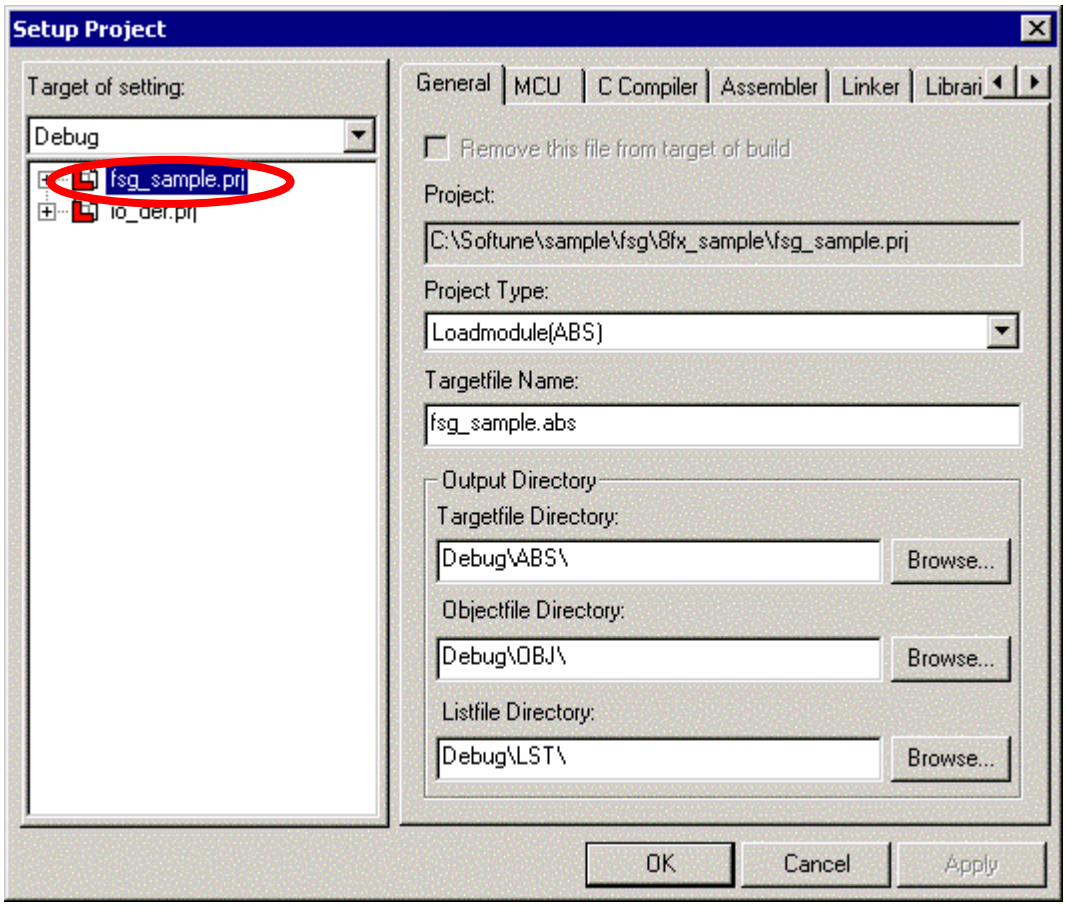
- Include path of Compiler
- Disposition/connection and register bank in Linker

《 Setting procedure 》

(1) Select [Setup Project] from the [Project] menu.

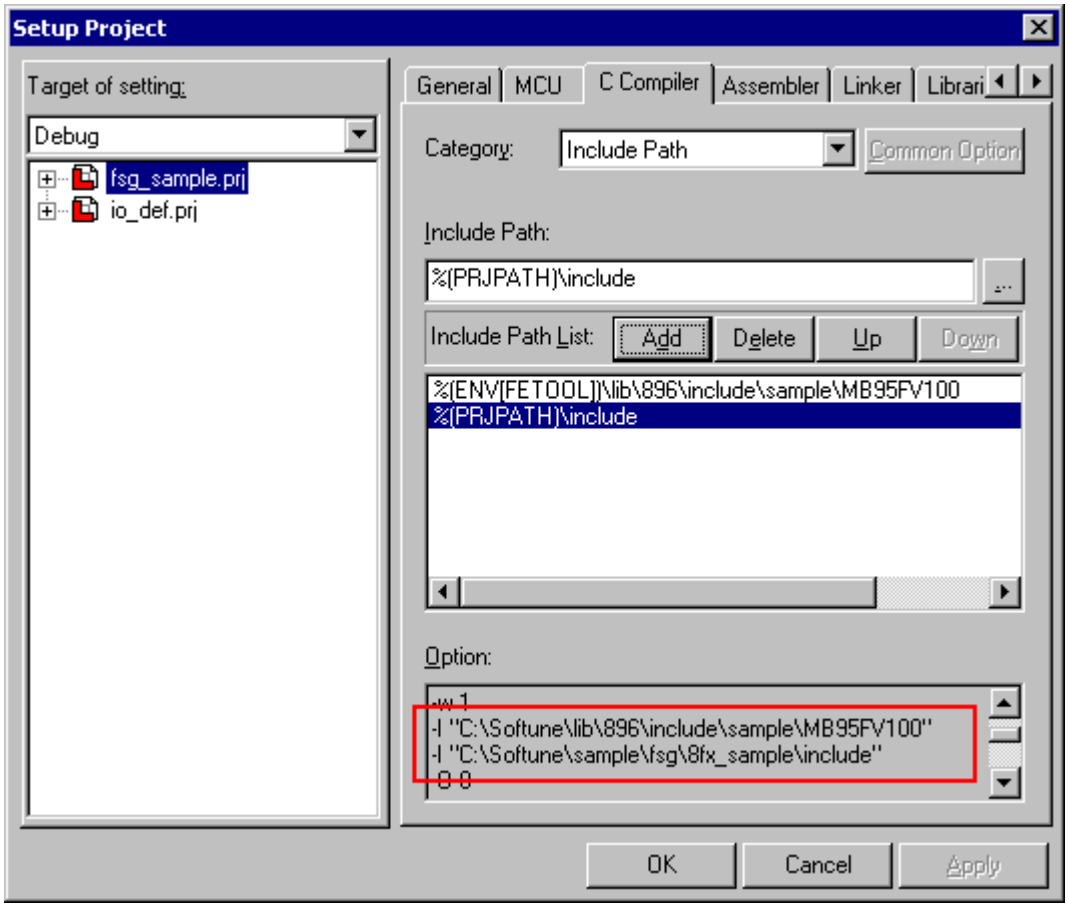


(2) Select **`fsg_sample.prj`** as the Project to be set.



(3) Select [Include Path] at Category in the C Compiler tab and set the include path. Click the [Add] button to add the path to [Include Path List].
The path is usually an absolute path. As shown below, this tutorial uses the macros for setting to easily move or copy the Project.
The descriptions using the macro are displayed in the Option column. Check that the post-expansion path is correct.

%(ENV[FETOOL])\lib\896\include\sample\MB95FV100 → ***SOFTUNE*** install directory
%(PRJPATH)\include → Project directory



One Point !

= What other macro can we use? =

%(FILE)	Transferred as full-path name of file
%(LOADMODULEFILE)	Transferred as full-path name of load module file
%(PRJFILE)	Transferred as full-path name of project file
%(WSPFILE)	Transferred as full-path name of workspace file
%(ABSPATH)	Transferred as target file directory
%(OBJPATH)	Transferred as object file directory
%(LSTPATH)	Transferred as list file directory
%(PRJCONFIG)	Transferred as project configuration name
%(ENV[environment variable])	Transferred as environment variable value specified by environment variable in brackets

For details, refer to Help (***Section 1.11 Macro Description Usable in Manager of SOFTUNE Workbench User's Manual***).

(4) Select [Disposition/Connection] at Category in the Linker tab and specify the ROM/RAM area.
By default, Mode 2 is set for Automatic Disposition and ROM/RAM Area List is also set according to a product.
In this sample, add the area (ROM_AREA) to allocate the CODE section to ROM/RAM Area List and change the start address of the area (INROM01) to allocate the CONST, @INIT, and @DIRINIT section.
When the section name is prefixed by "@", the ROM address of the ROM-to-RAM transfer section is specified that transfers data from ROM to RAM during execution.

One Point !

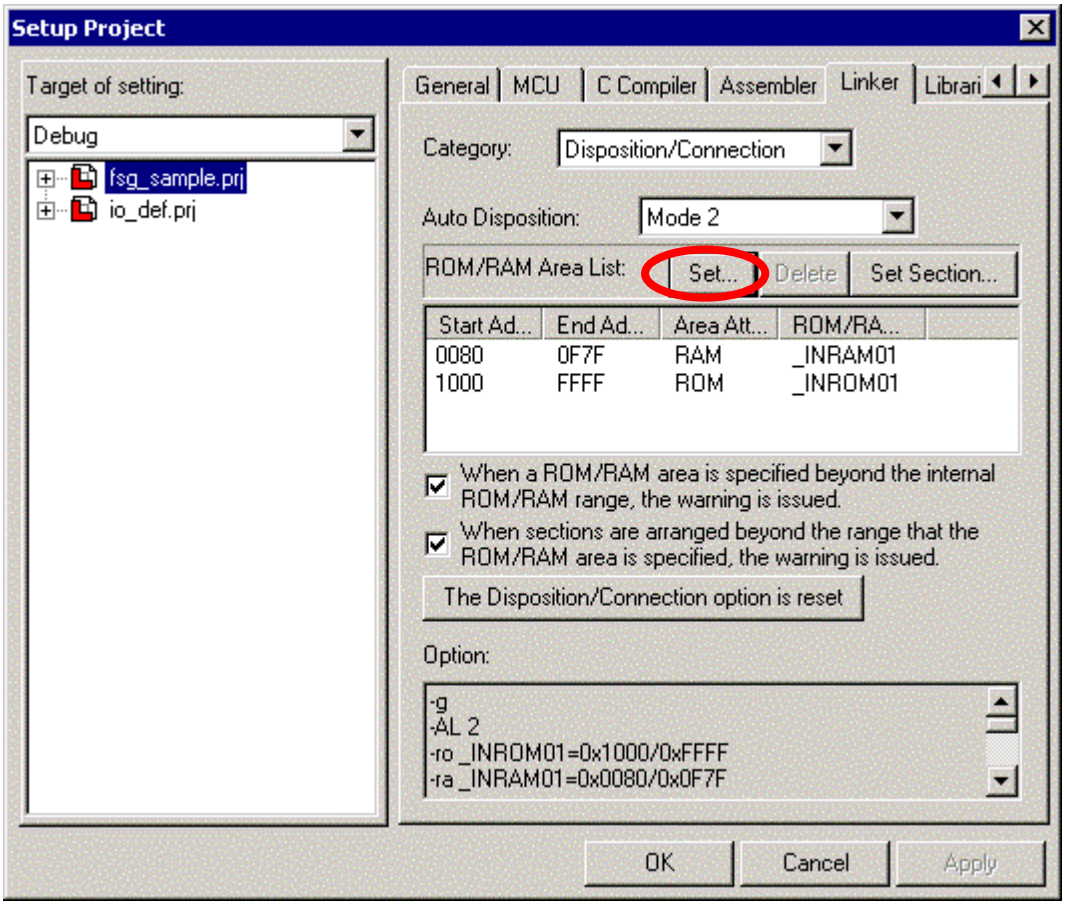
= What is the ROM-to-RAM transfer section? =

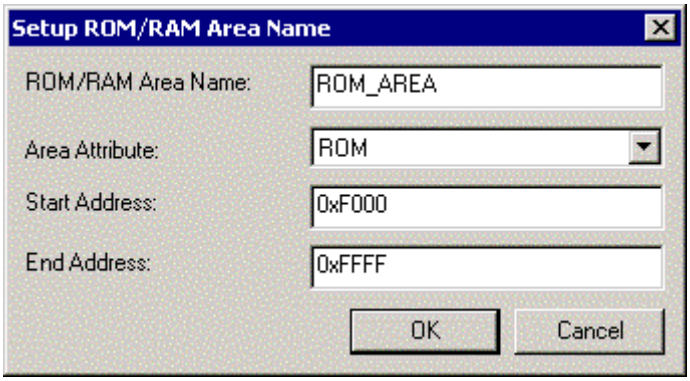
In developing a program using C Compiler, variables with initial values are generated. Because these variables are rewritten during execution an application, they must be in RAM. Therefore, in built-in programs, initial value data cannot be used unless it is placed in ROM and is transferred to RAM before executing the application.

The ROM-to-RAM transfer section enables such use of initial value data.

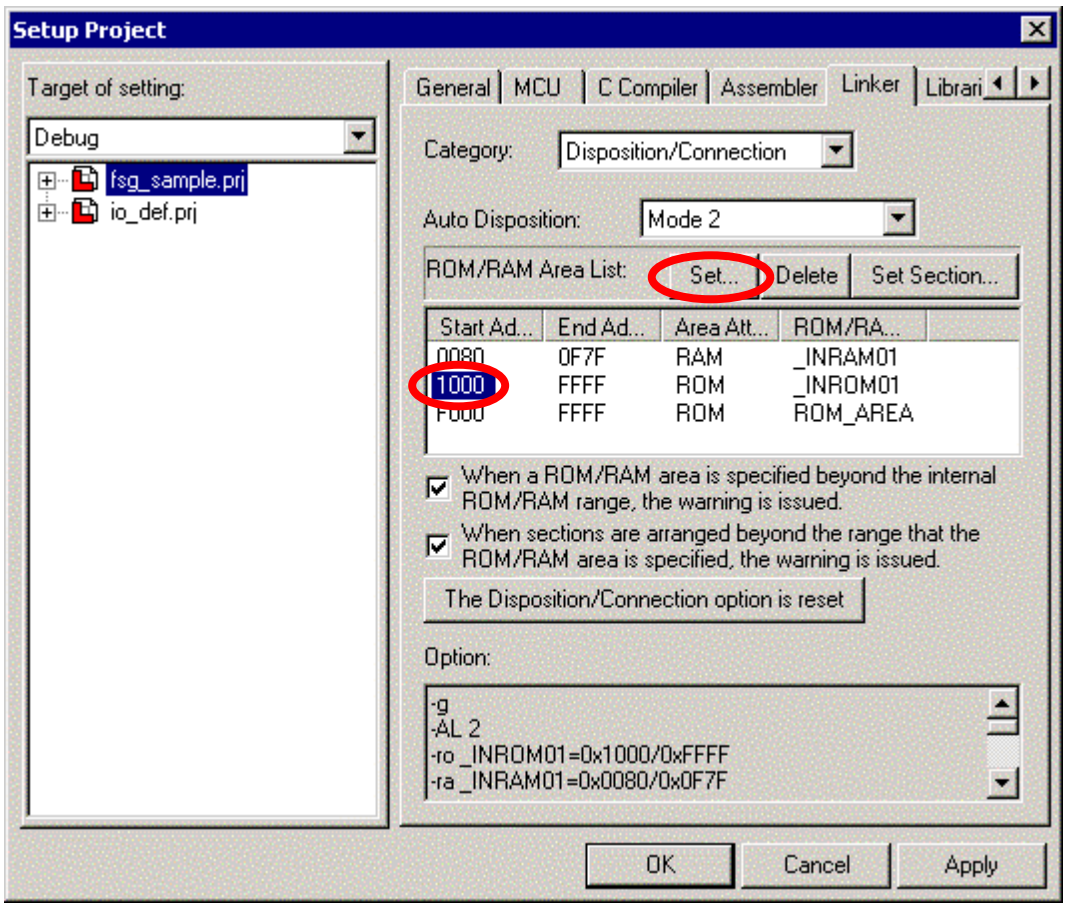
For details, refer to Help (***Section 5.9 Sections to be transferred from ROM to RAM of SOFTUNE Linkage Kit MANUAL.***)

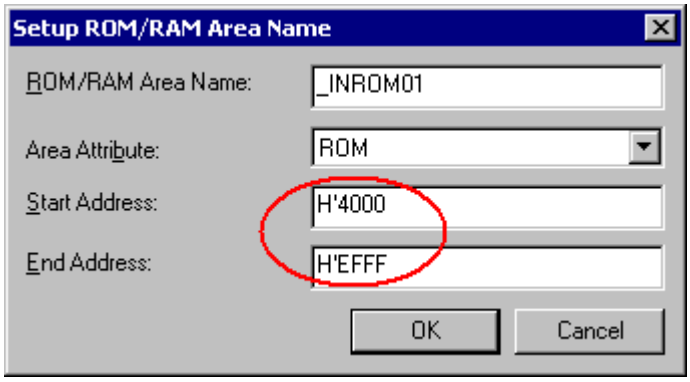
Click the [Set…] button and set the area name, attribute, and start and end addresses of the ROM or RAM area in the [Setup ROM/RAM Area Name] dialog.



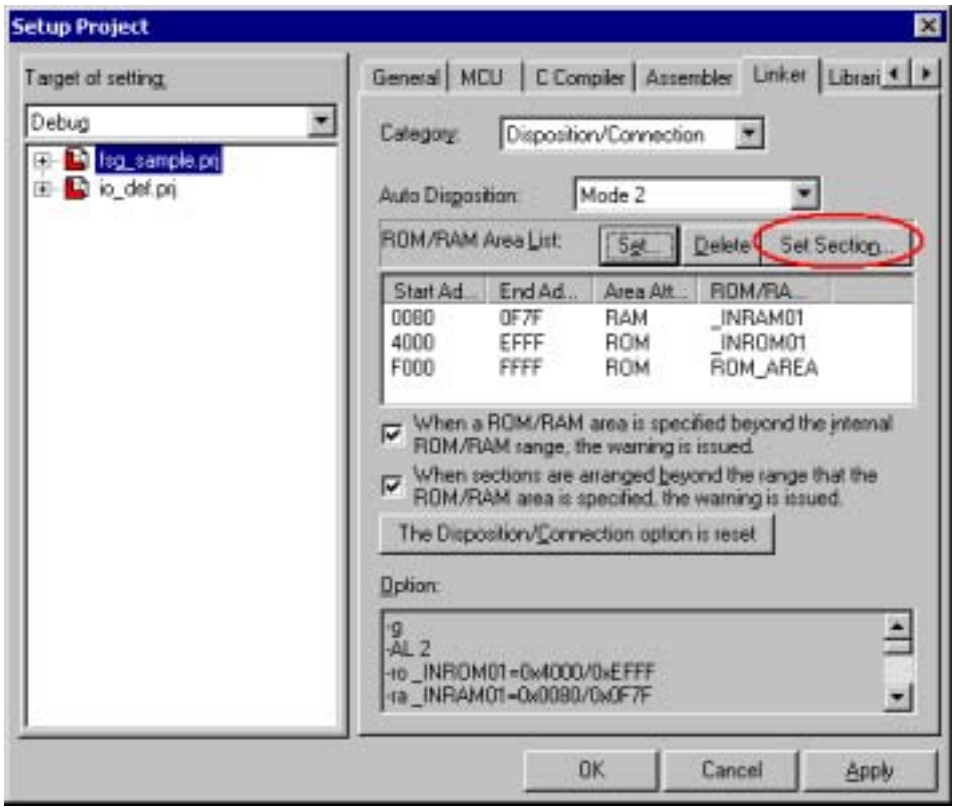


Select the start address of the _INROM01 and click the [Set..] button to change the start address in the [Setup ROM/RAM Area Name] dialog to H'4000. And change the end address to H'EFFF.

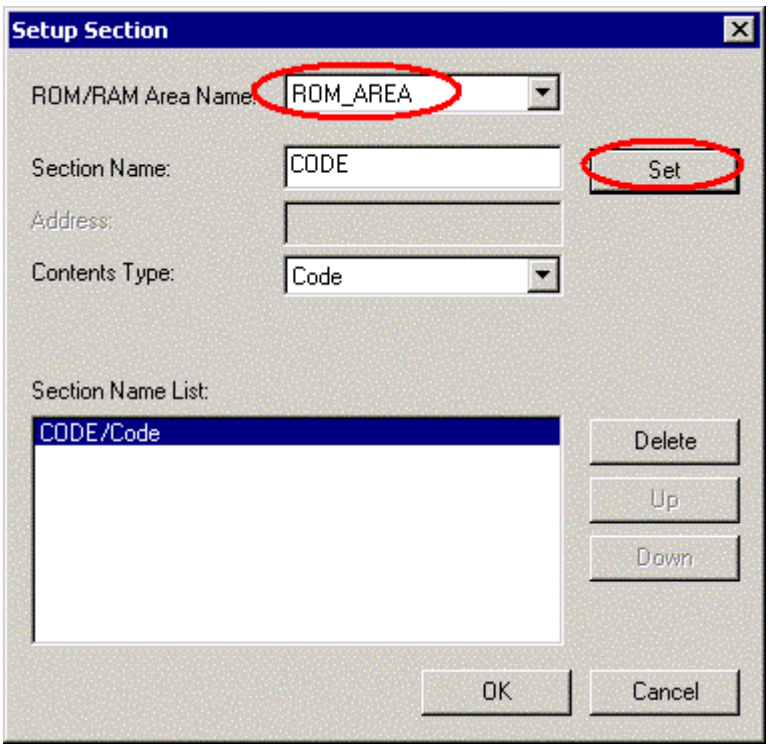


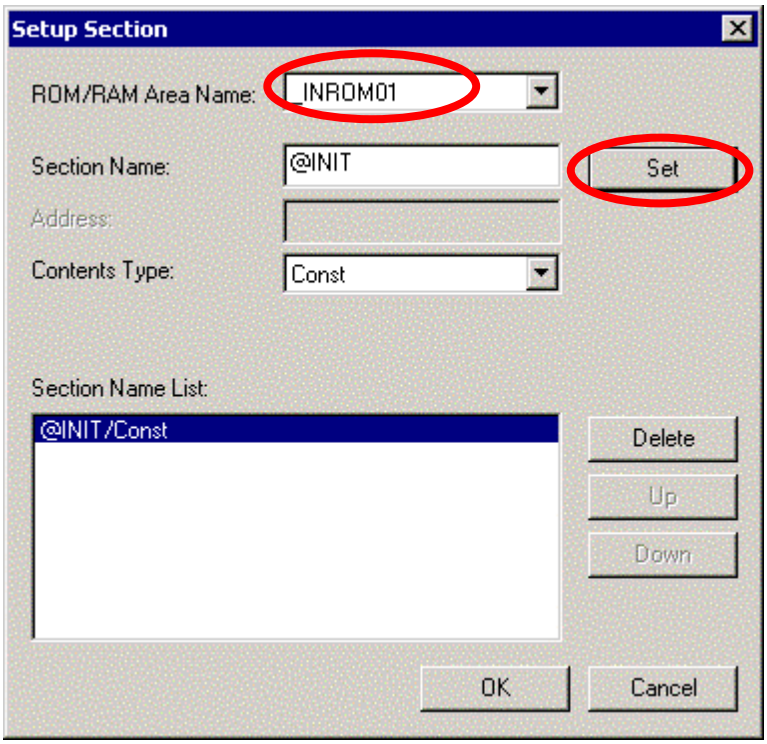


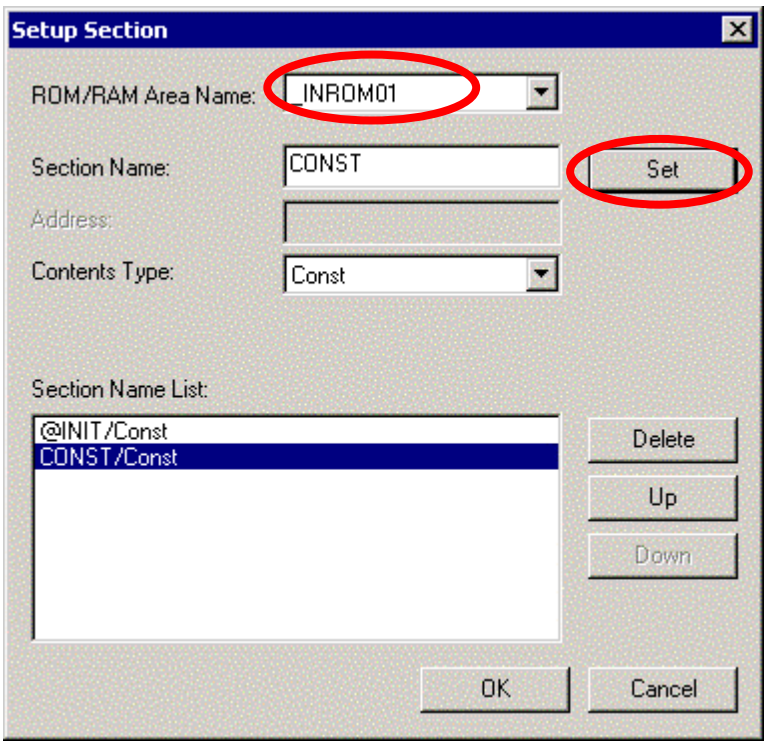
Click the [Set Section...] button in the ROM/RAM Area List and set the area to allocate the CONST, @INIT or @DIRINIT section.

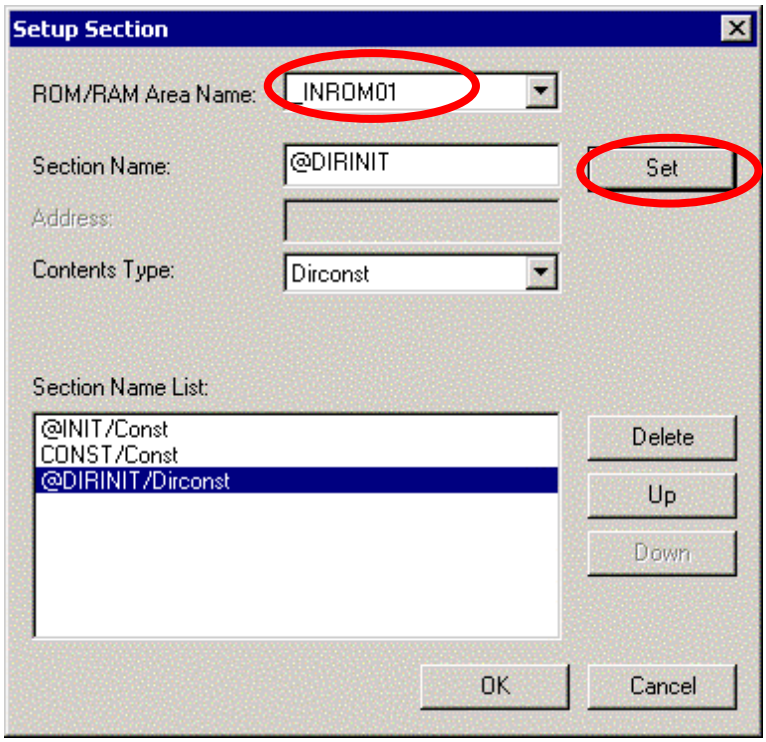


Click the [Set] button to add the section name to Section Name List. After confirming that the section name is added, click the [OK] button.

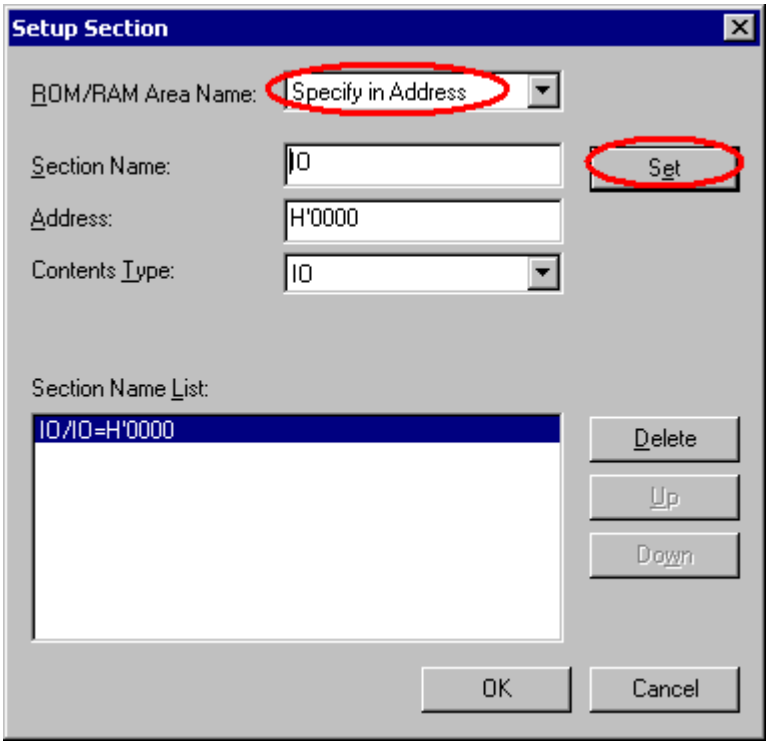



Setup Section
ROM/RAM Area Name:
_INROM01
Section Name:
@INIT
Set
Address:
Contents Type:
Const
Section Name List:
@INIT/Const
Delete
Up
Down
OK
Cancel

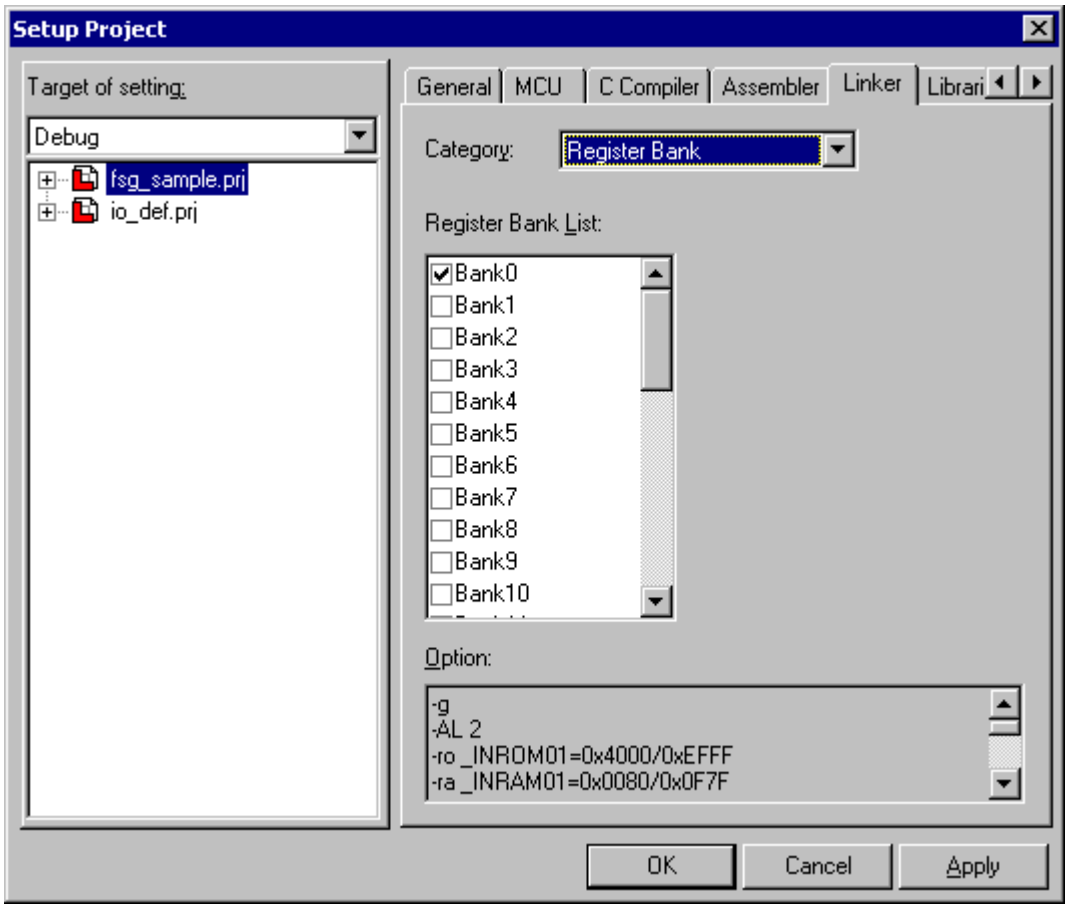

Setup Section
ROM/RAM Area Name:
_INROM01
Section Name:
CONST
Set
Address:
Contents Type:
Const
Section Name List:
@INIT/Const
CONST/Const
Delete
Up
Down
OK
Cancel



Set the IO section address so that the following warning may be avoided.
Click the *** W1373L: The section is placed outside the I/O area (IO)



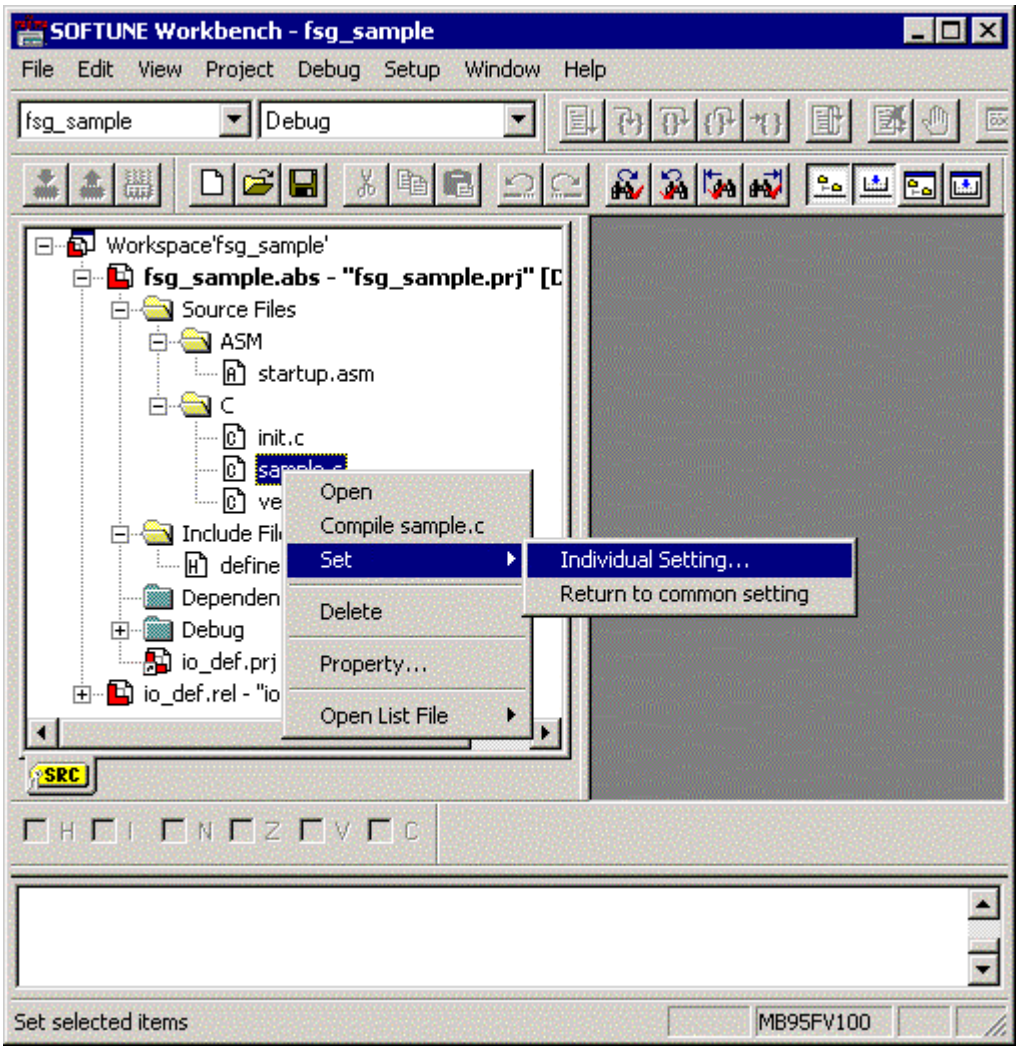
(5) Select Register Bank at Category in the Linker tab and set the register bank.
By default, the checkbox is marked only for Bank 0. Mark the checkboxes for all the banks to be used.
In this sample, click the [OK] button with the checkbox only for Bank 0 marked.



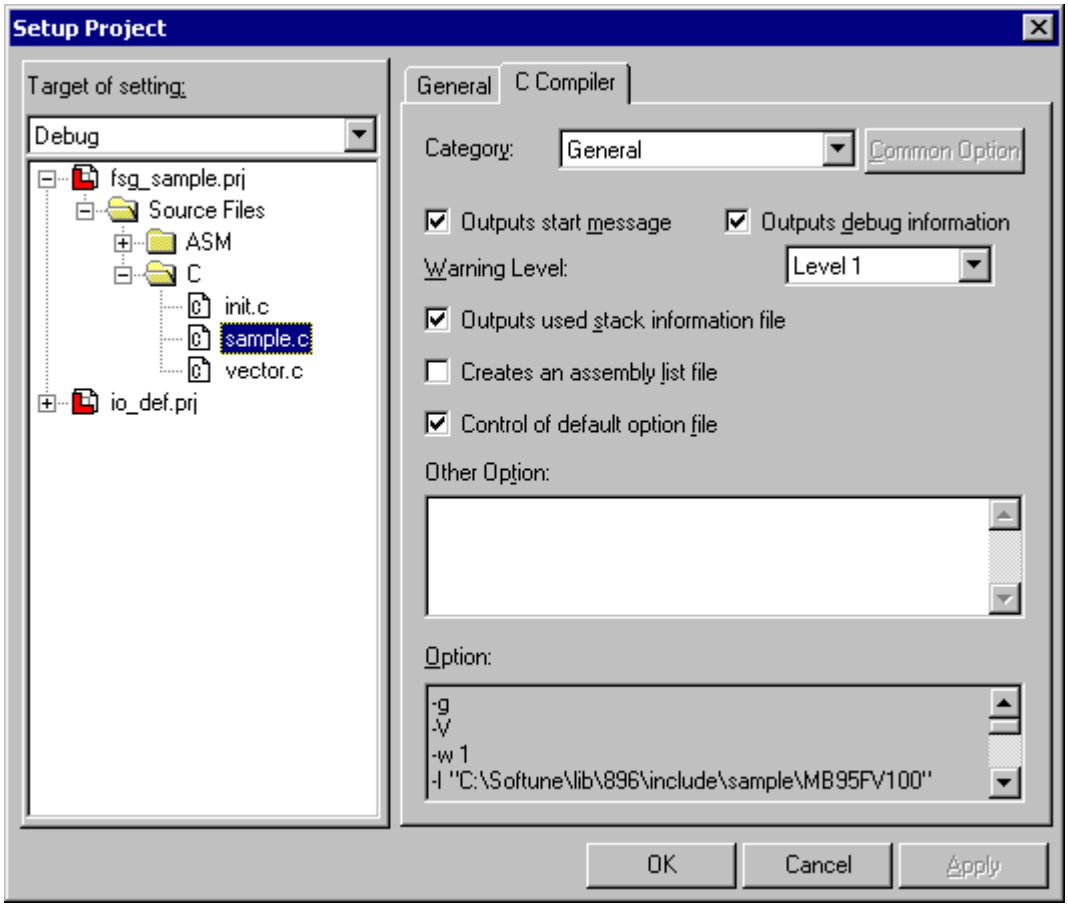
One Point !

= How to set language tool options for each source file =

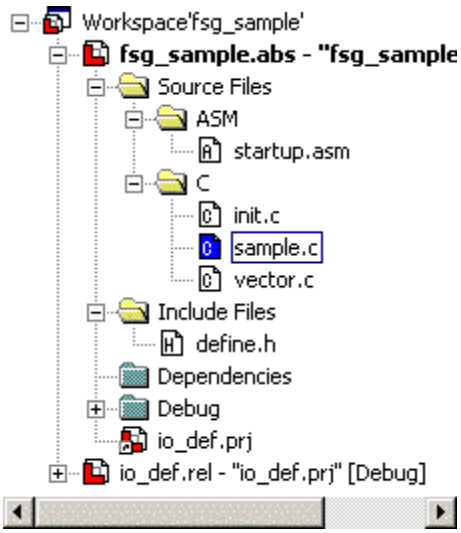
(1) Select the source file to be set individually in the Project window to display the shortcut menu. Select [Set]-[Individual Setting...] to set individual options in the Setup Project dialog.



(2) When the C source or the Assembler source is selected, the C Compiler tab or the Assembler tab is displayed. Set each item.



(3) Click the [OK] button.
The source file for which the options are individually set is displayed in blue.



4.1.7 Making Target File

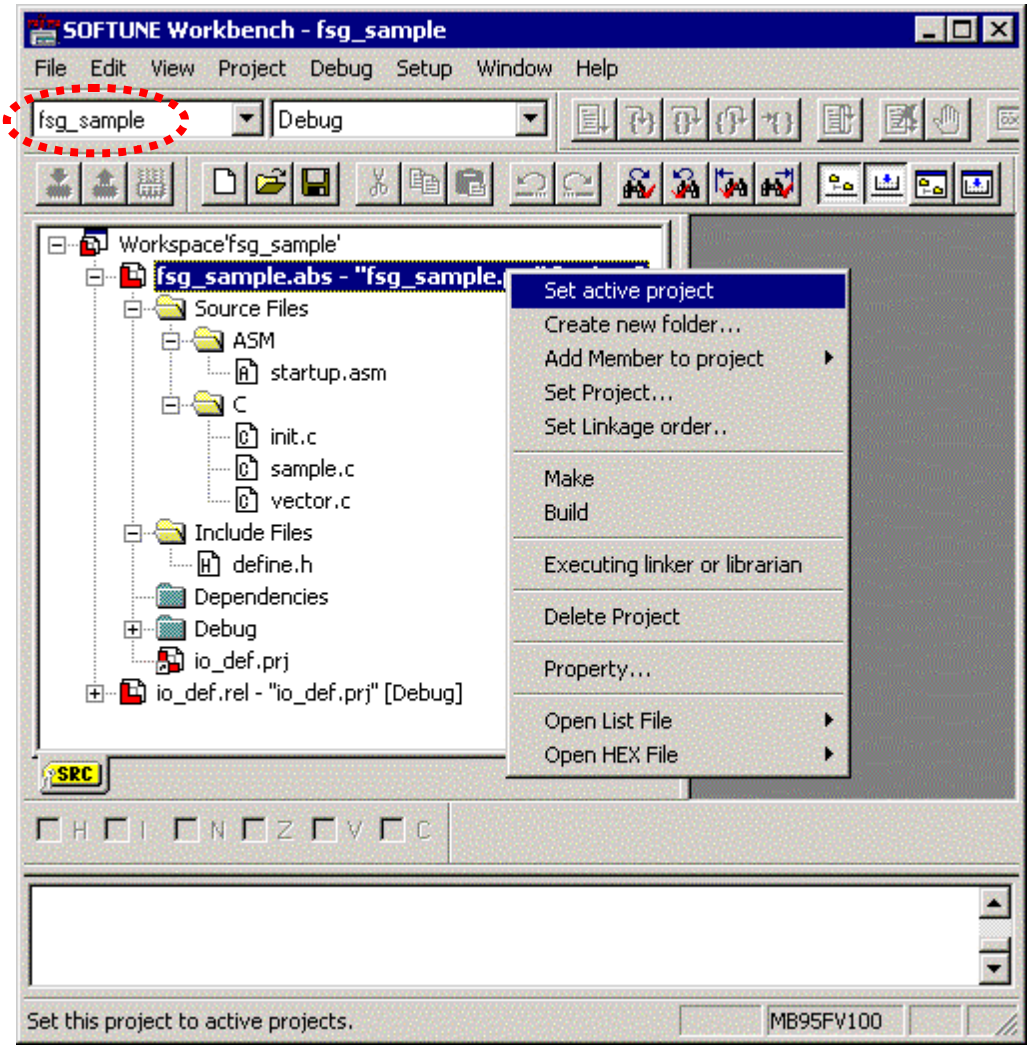
■ **Create Target File**

After setting the project for Make or Build in the active project, execute Make or Build to make a target file. When Make or Build is executed, the include file called from the source file is retrieved and dependencies are updated automatically.

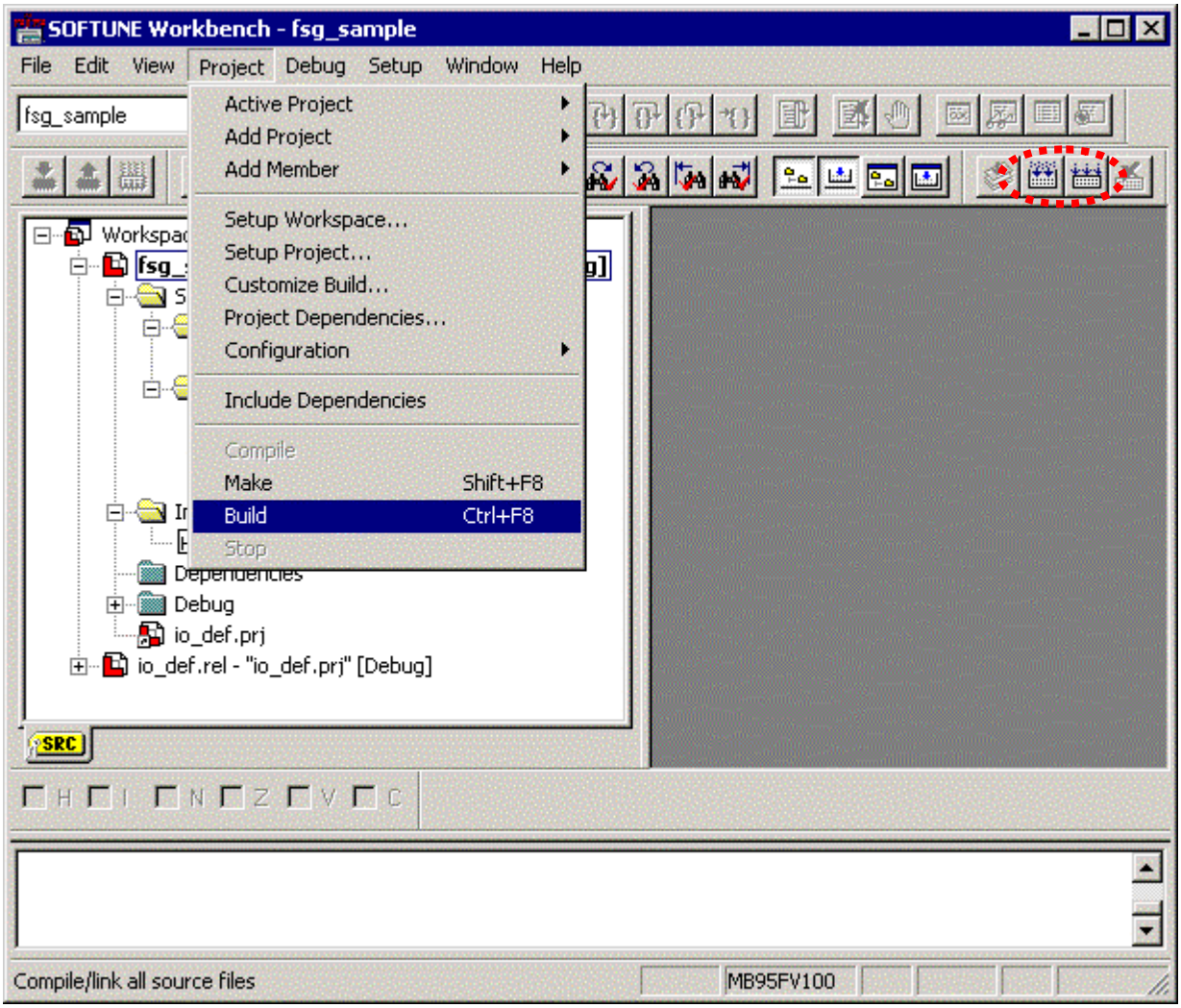
《 Operating procedure 》

(1) Set **`fsg_sample.prj`** to the active project.

Select **`fsg_sample.prj`** in the Project window and select [Set Active Project] from the shortcut menu. Selecting fsg_sample enclosed in the dotted lines also enables this setting.



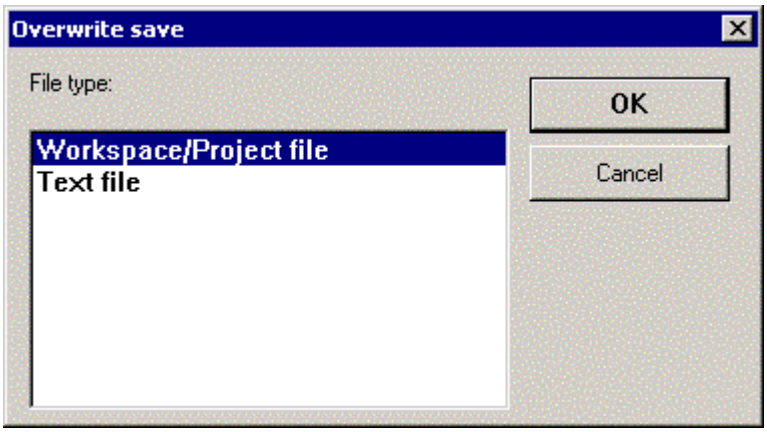
(2) Select [Make] or [Build] from the [Project] menu.
Clicking the button (enclosed in the dotted lines) also enables [Make] or [Build].



(3) Save the Project file.

The project file is not saved automatically until the workspace is closed. When updated, save the project file.

Select the [Save...] command from the [File] menu and choose the Project file in the [Overwrite Save] dialog.



One Point !

= Difference between Make and Build =

- Make

  This function compiles/assembles only the updated source file, and then connects all objects and libraries to make a target file.

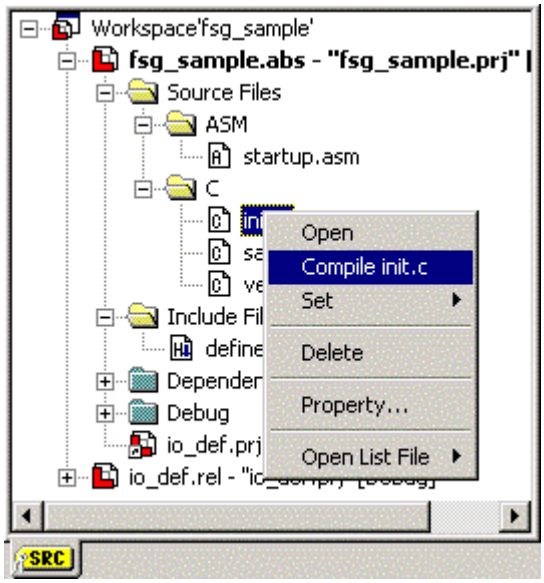
- Build

  This function compiles/assembles all source files stored in the Project, and then connects all objects and libraries to make a target file.

One Point !

= Procedure for Compiling/assembling =

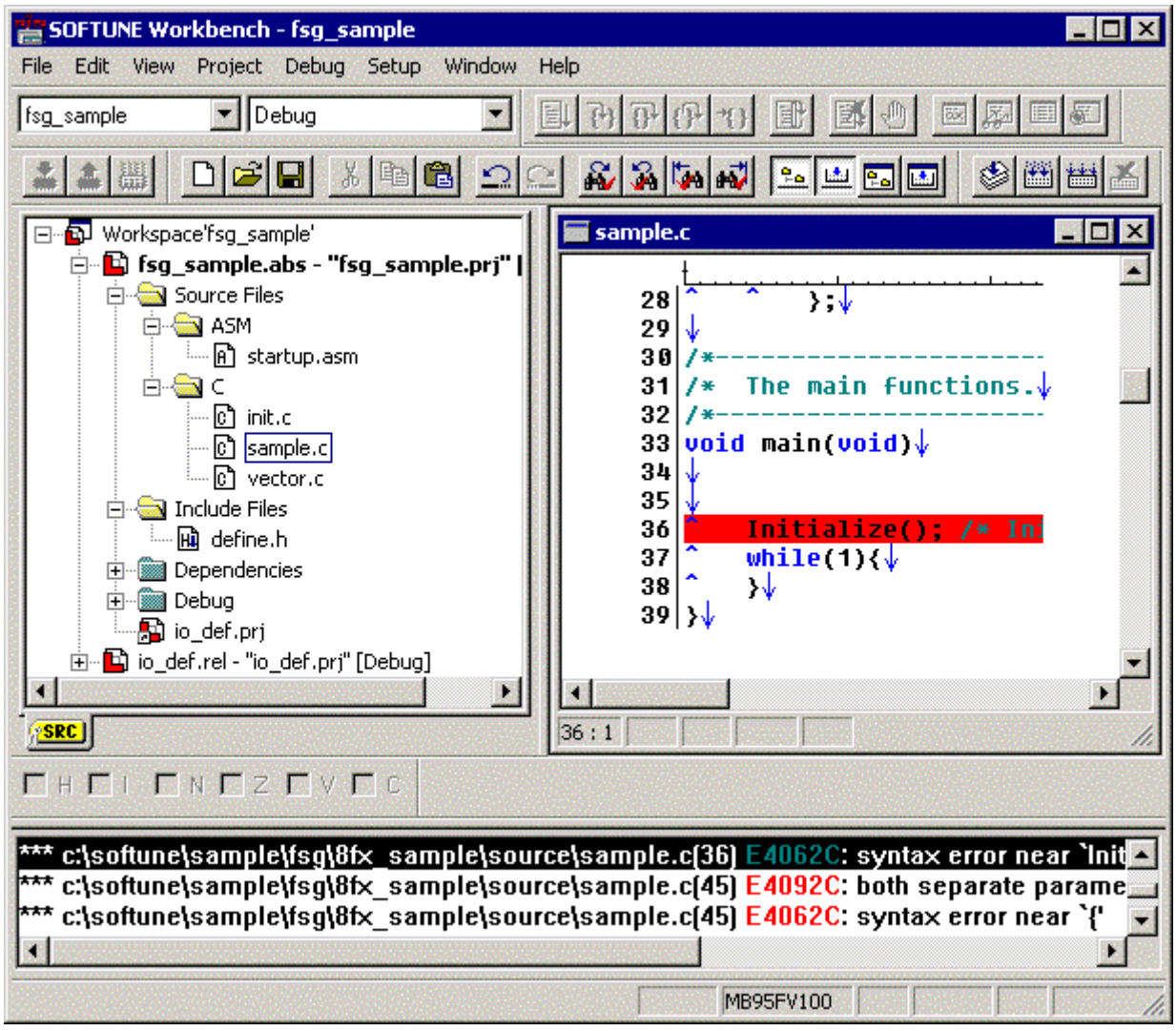
Any one of source files can be compiled or assembled.
Select the source file to be compiled in the Project window and execute [Compile] or [Assemble] in the shortcut menu.



One Point !

= Error jump function =

SOFTUNE Workbench has an error jump function.
When an error occurs in compiling or assembling, the error message is displayed on the Output window.
When the user double-click the error message, the source file starts up to jump to the error line.



= Using external editor =

When the user uses an external editor, the source file being edited is not saved automatically. Therefore, the user must save the source file before compiling.

In addition, when the user must set different options for each editor to use the error jump function. For details, refer to Help (***Section 1.9 Storing External Editors of SOFTUNE Workbench User's Manual***).

■ **Recommended Use of C Checker**

Fujitsu recommends that the user should use the C Checker after a compiling error in the source file is corrected.
Using the C Checker enables checking of bug, portability and coding errors that are not treated as compiling errors .
For details, see ***SOFTUNE C Checker Manual***

■ **Recommended Use of C Analyzer**

Fujitsu recommends that the user should use the C Analyzer after making a target file.
Using the C Analyzer enables visual checking of the program structure.
For details, see ***SOFTUNE C Analyzer Manual***.

4.2 Debugging by Simulator

■ Debugging by Simulator

This section explains the procedure for verifying program by using the Simulator. It also explains briefly the Simulator functions.

- 4.2.1 Making Setup Wizard for Simulator
- 4.2.2 Setting Memory Map
- 4.2.3 Setting Interrupt
- 4.2.4 Registering and Checking Variables
- 4.2.5 Setting Breakpoint
- 4.2.6 Executing and Stopping Program
- 4.2.7 Mix Display
- 4.2.8 Monitoring
- 4.2.9 Correcting and Re-debugging Program

4.2.1 Making Setup Wizard for Simulator

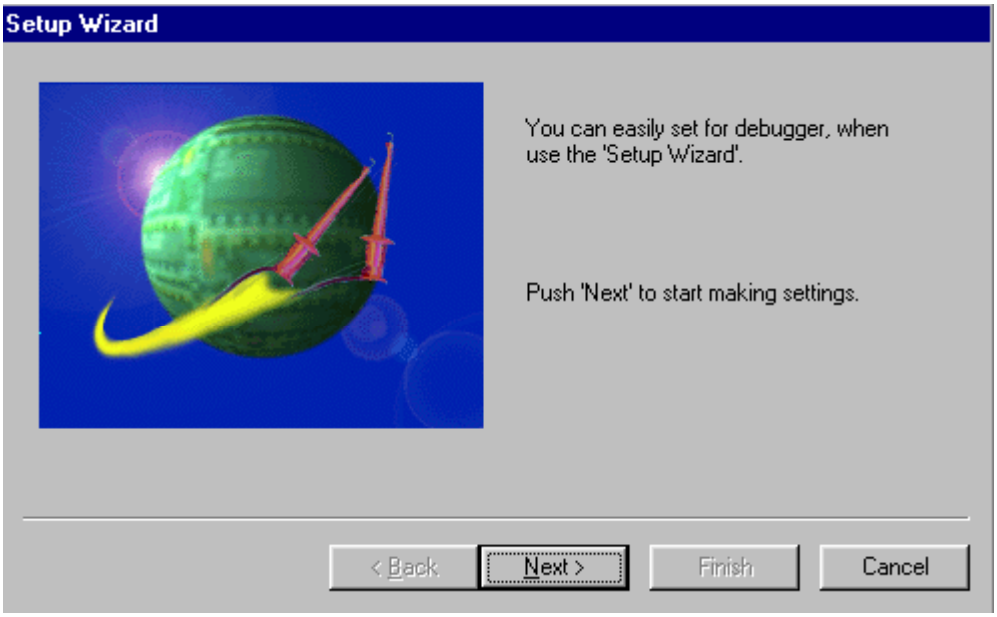
■ Setting Procedure by Setup Wizard

When debugging for the first time after making Project, execute [Start debug] in the [Debug] menu to start the Debugger setup wizard. Set each item according to the menu.

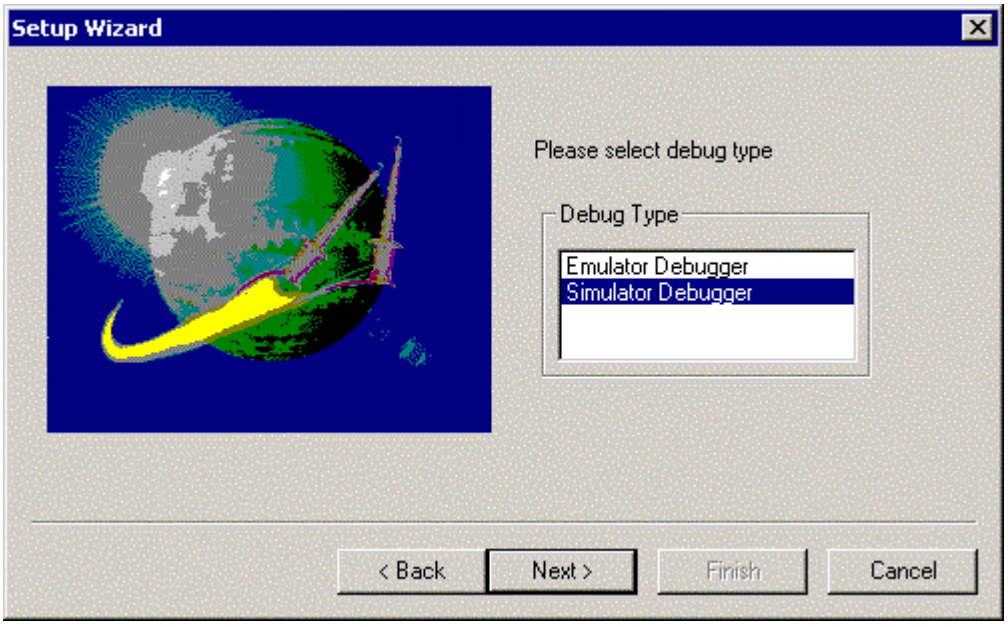
《 Setting procedure 》

(1) Select [Start debug] from the [Debug] menu.

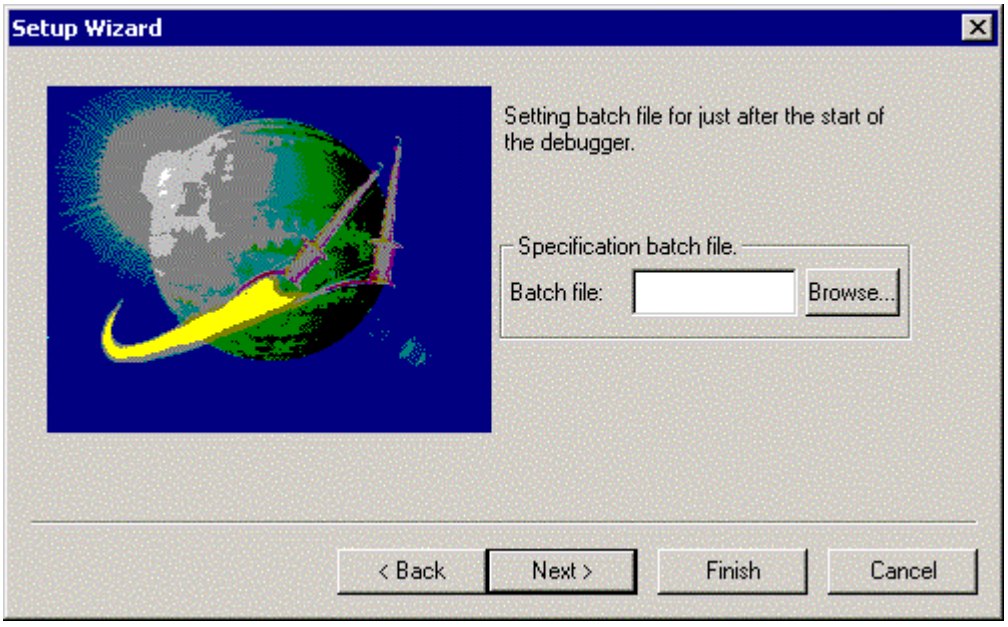
When the setup wizard is started, click [Next].



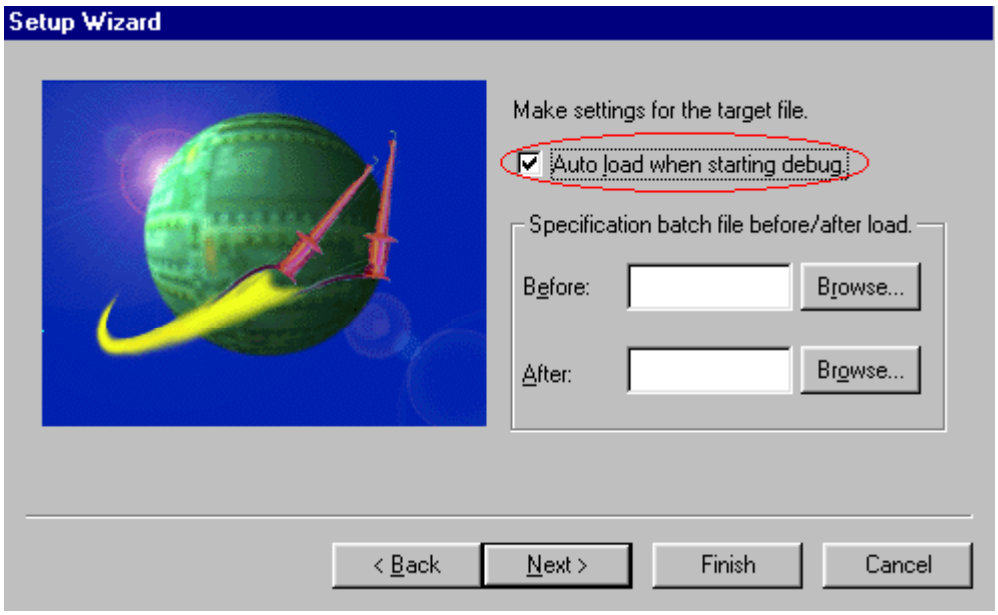
(2) Select [Simulator Debugger] at Debugger Type.



(3) Click [Next] without specifying Batch File.

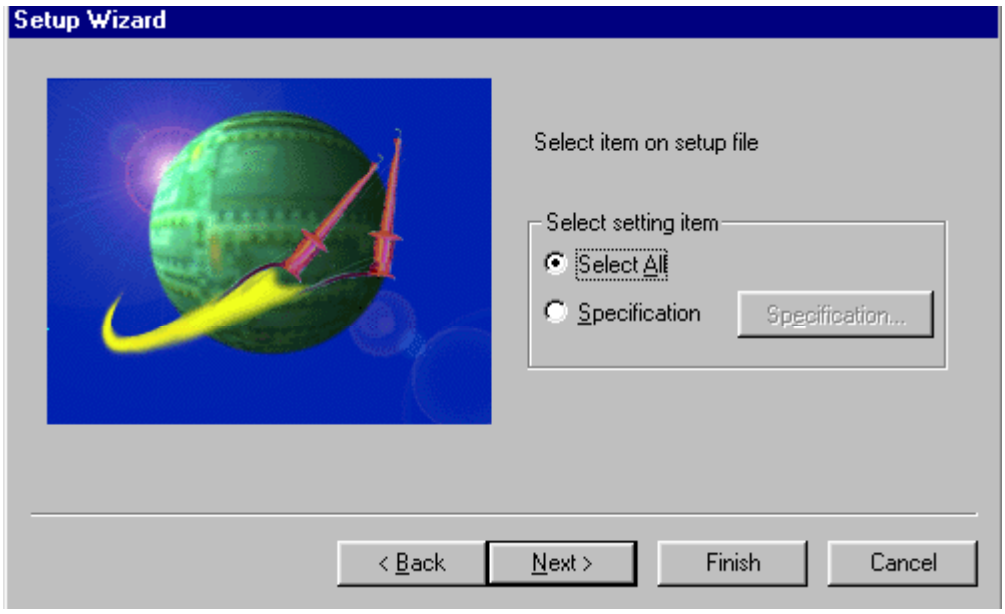


(4) Mark the checkbox for [Auto load when starting debug]. The target file is automatically loaded when debugging is started.

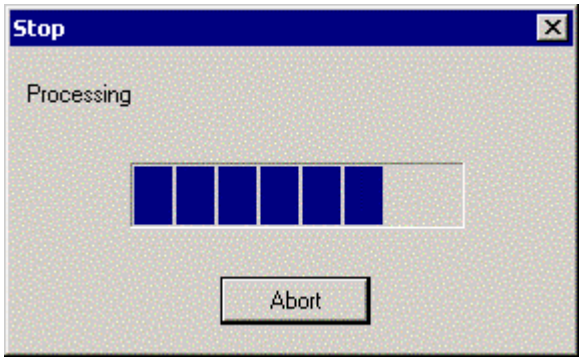


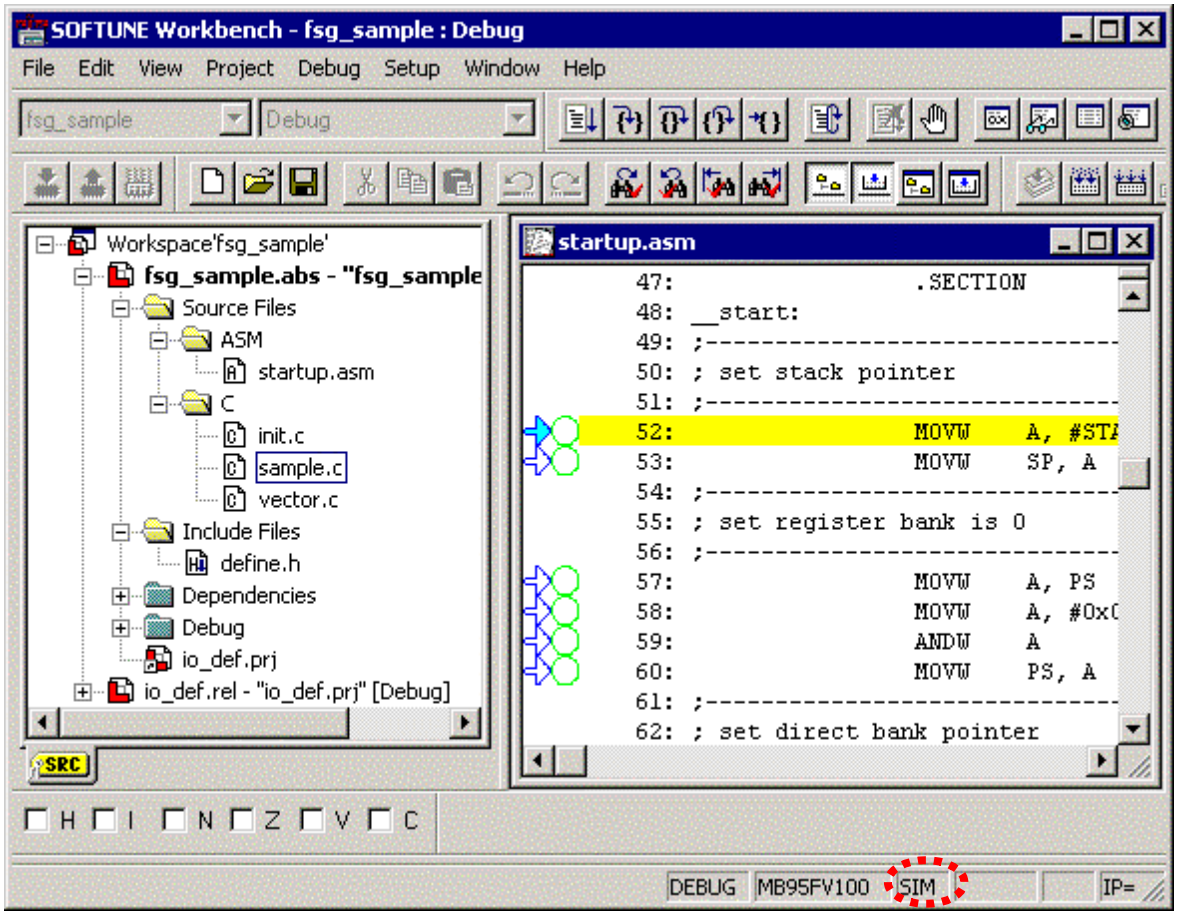
(5) Set items about saving debugging environment.

Select [Select All] for Select setting item.



(6) Follow the instructions on the screen and click the [Finish] button. This completes the settings.
The target file is loaded to switch to the debug mode.
The type of debug modes appears at the bottom of the screen (enclosed in dotted line).





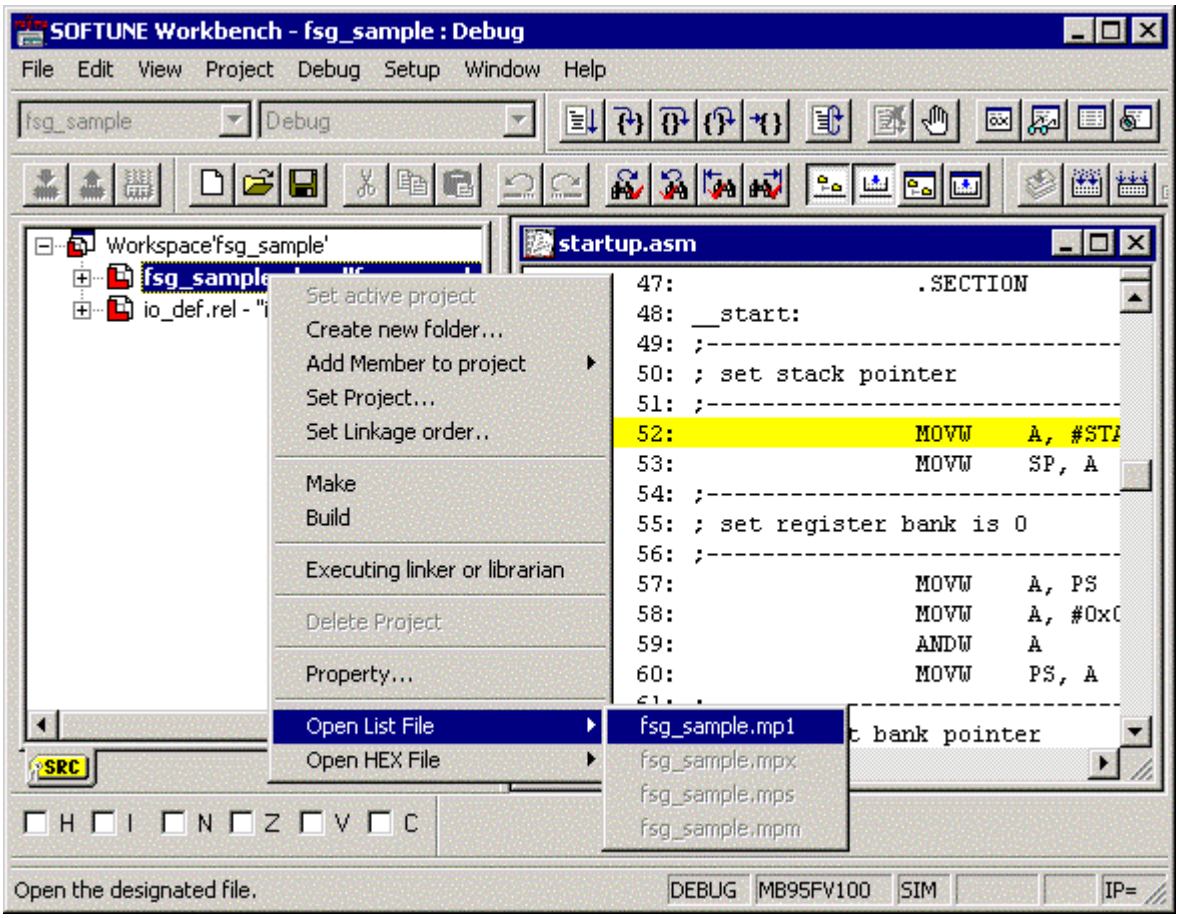
4.2.2 Setting Memory Map

Set the memory map.
If program access is made with anything other than the set attribute, an access violation occurs to stop the program.
In this sample, set the area to access the CONST, #INIT, CODE, INITVECT, and RESETVECT section.

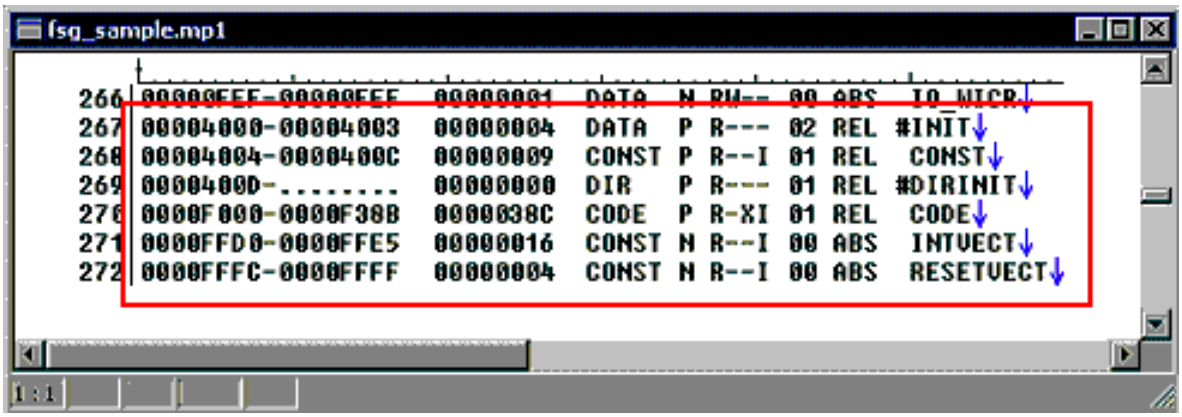
《 Setting procedure 》

(1) Check the CONST, #INIT, CODE, INITVECT, and RESETVECT sections on the map file.

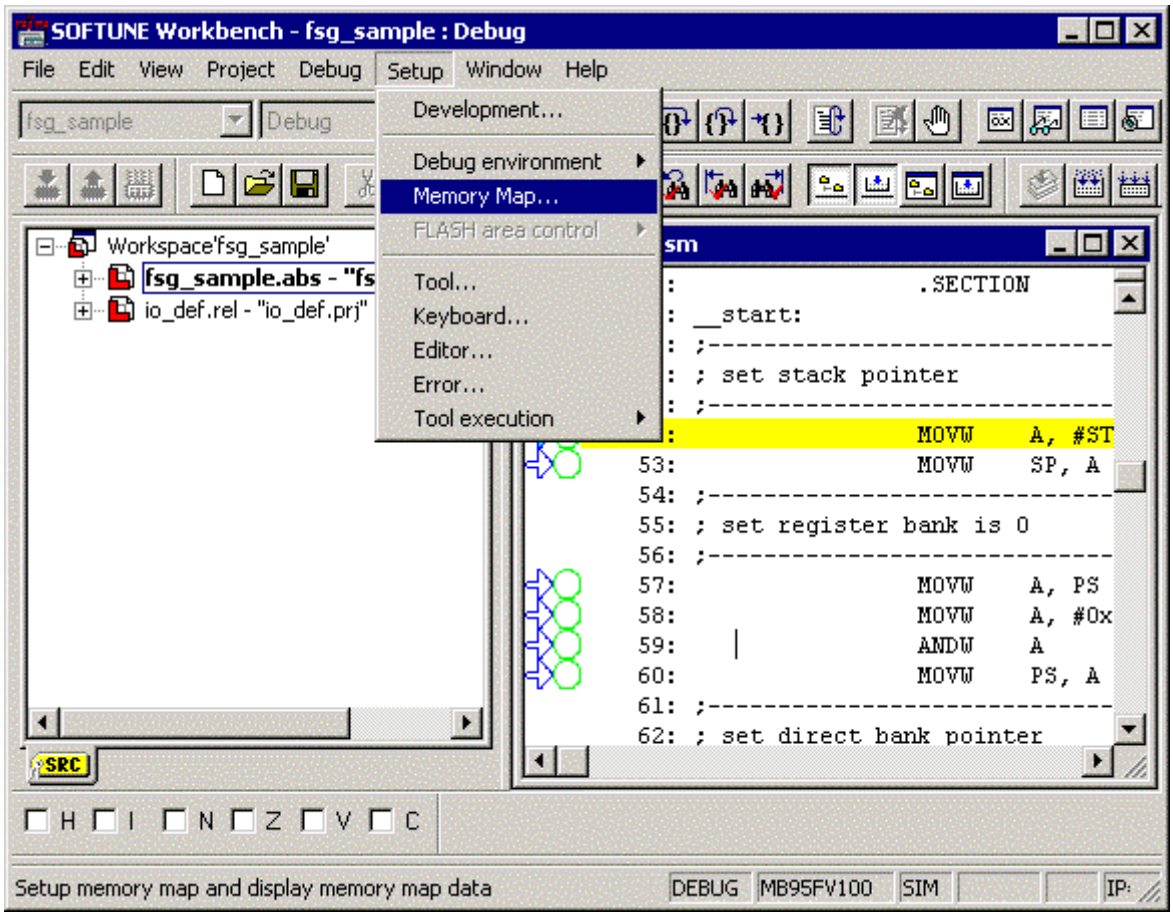
To open the map file, select "**fsg_sample.abs**" as the project name and select [Open list file]-[fsg_sample.mp1] from the shortcut menu.



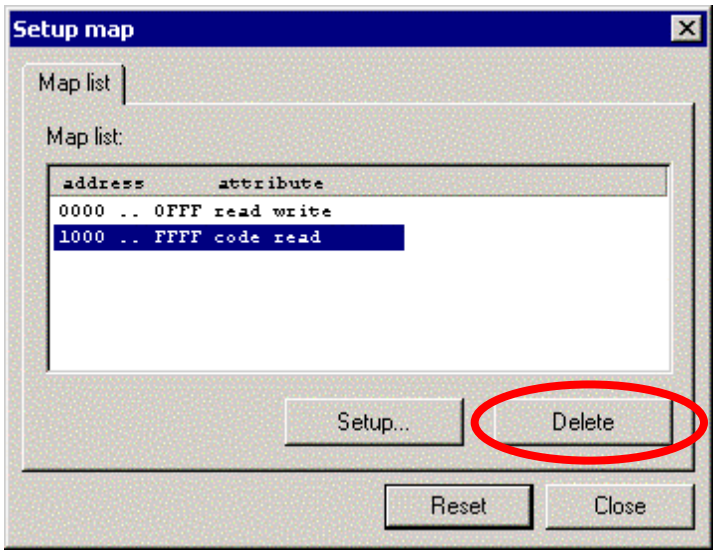
As can be seen from the enclosed part, CONST and #INIT are at addresses 4000 to 400C, CODE at addresses F000 to F38B, INITVECT at addresses FFD0 to FFE5, and RESETVECT at addresses FFFC to FFFF.



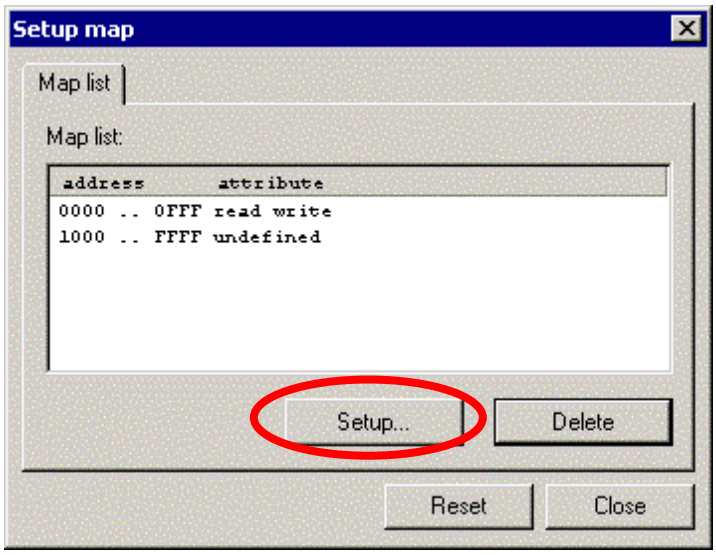
(2) Select [Memory Map] from the [Setup] menu.



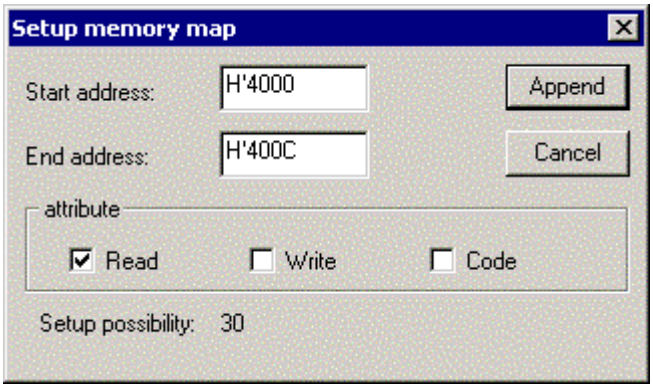
(3) Specify the map area (1000..FFFF) in the Map List tab and click the [Delete] button.



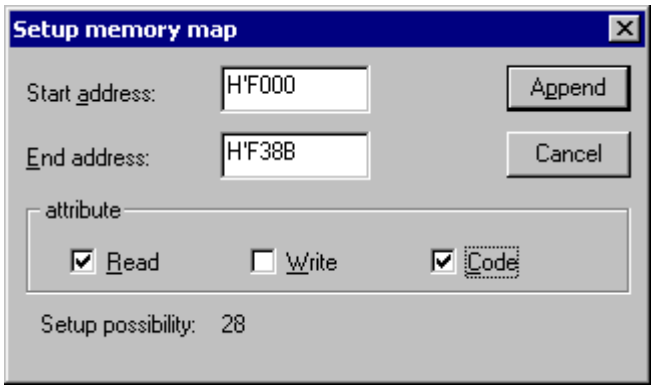
(4) Click the [Setup...] button in the Map List tab.



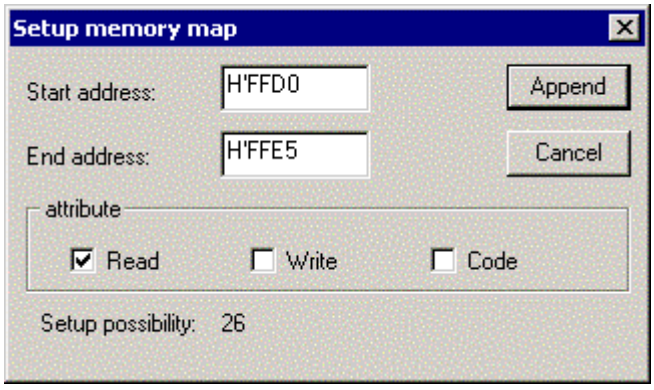
(5) Set the area to access the CONST and #INIT sections.



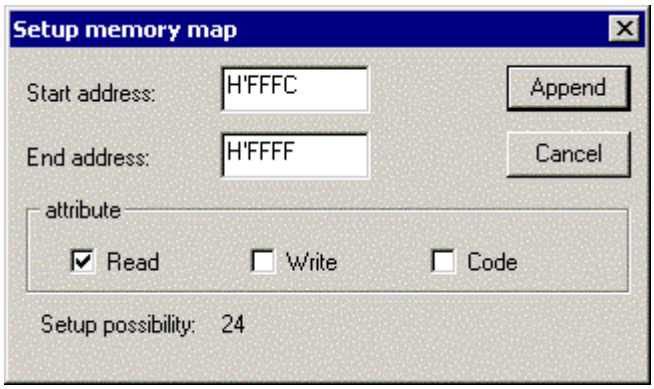
(6) Set the area to access the CODE section.

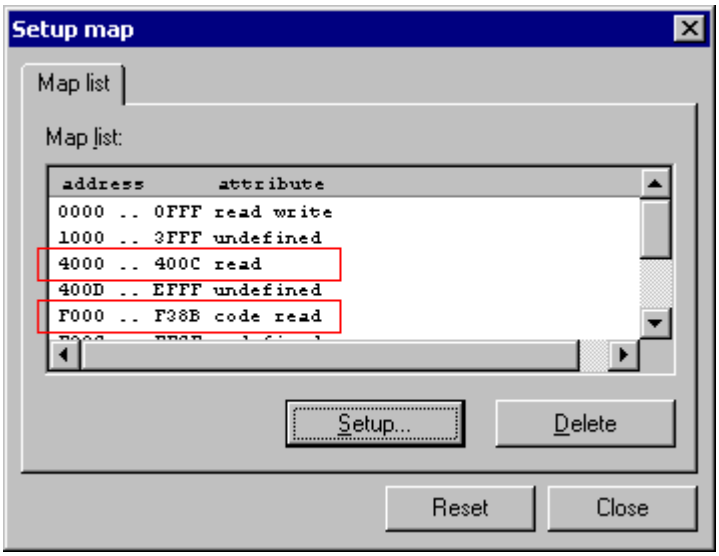


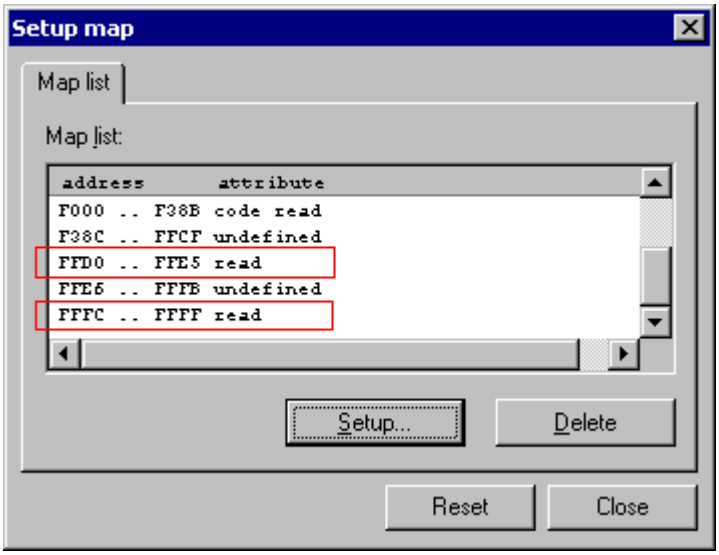
(7) Set the area to access the INITVECT section.



(8) Set the area to access the RESETVECT section.







= Delete defined areas =

Undefined areas may be changed into **defined** areas (read/write/code). However, **defined** areas cannot be changed into **undefined** areas in the [Setup map] dialog.

To change into undefined areas, delete **defined** areas.

4.2.3 Setting Interrupt

■ **Simulation of Timer Interrupt**

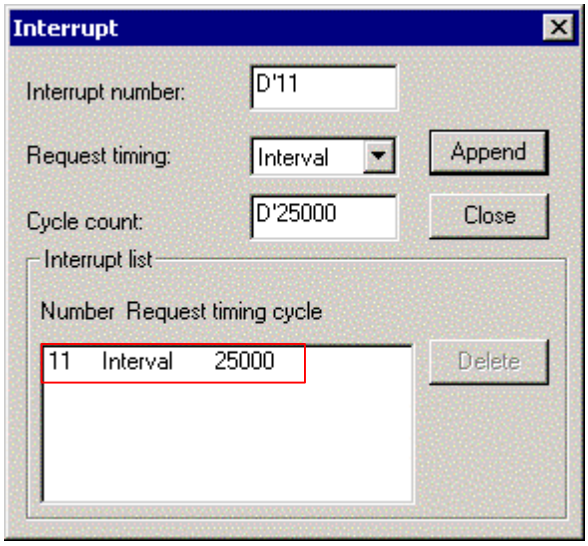
This sample program generates interrupts every fixed period of time using the timer resource.
However, because the Simulator has no timer resource, the user needs to set to generate specified interrupts within a fixed cycle in order to simulate the interrupt processing.
This setting is saved after debugging.

《 Setting procedure 》

(1) Select [Interrupt] from [Debugging environment] in the [Setup] menu.

(2) Set the interrupt number (vector number), request timing, and cycle count in the [Interrupt] dialog.

In this sample, set these items as follows.



4.2.4 Registering and Checking Variables

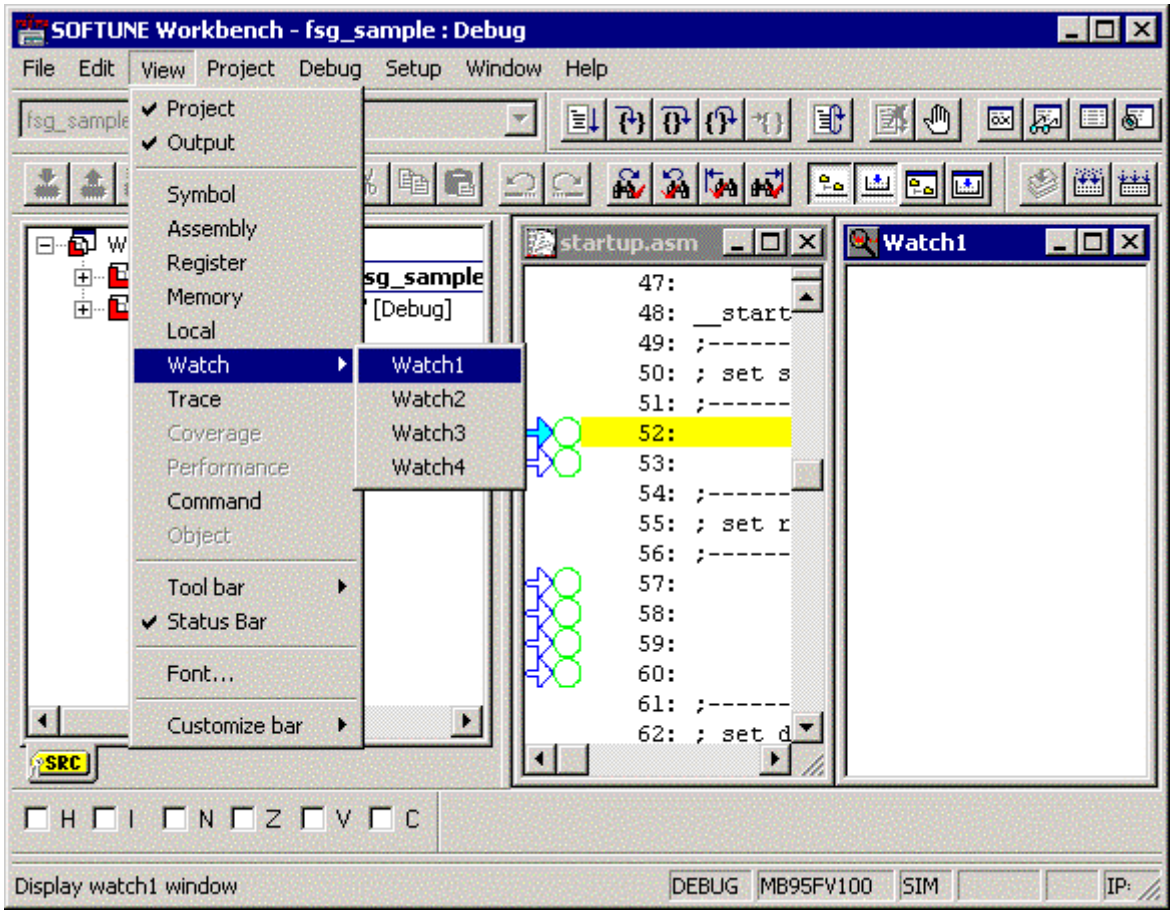
■ **Registering and Checking Variables**

Check the descriptions in the Watch window and Memory window.

《 Setting procedure 》

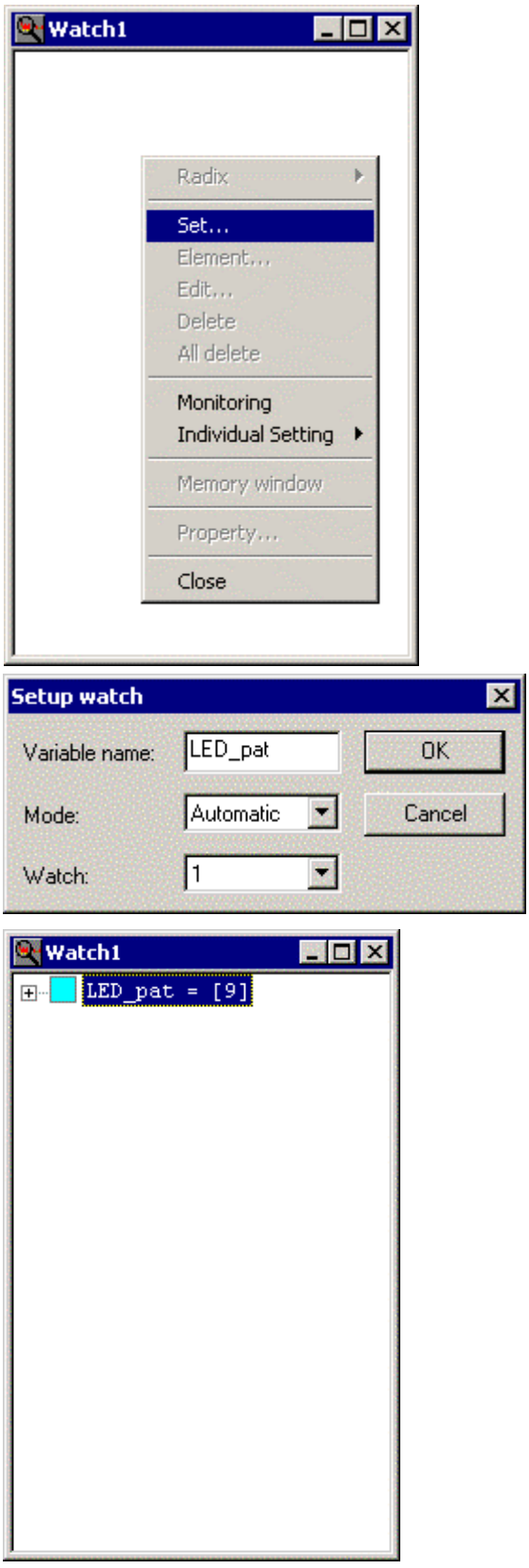
(1) Open the Watch window.

Select [Watch]-[Watch 1] from the [View] menu.

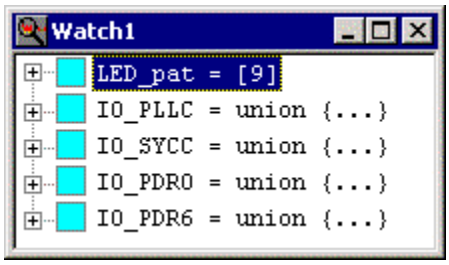


(2) Register variables.

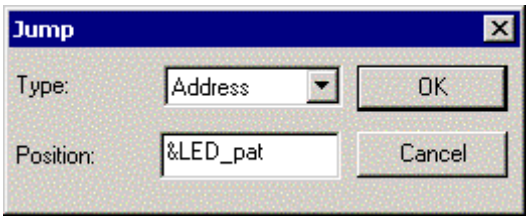
Right-click the Watch window to select [Set...] from the shortcut menu. Enter the variable name in the Setup Watch dialog and click the [OK] button to register the variables.



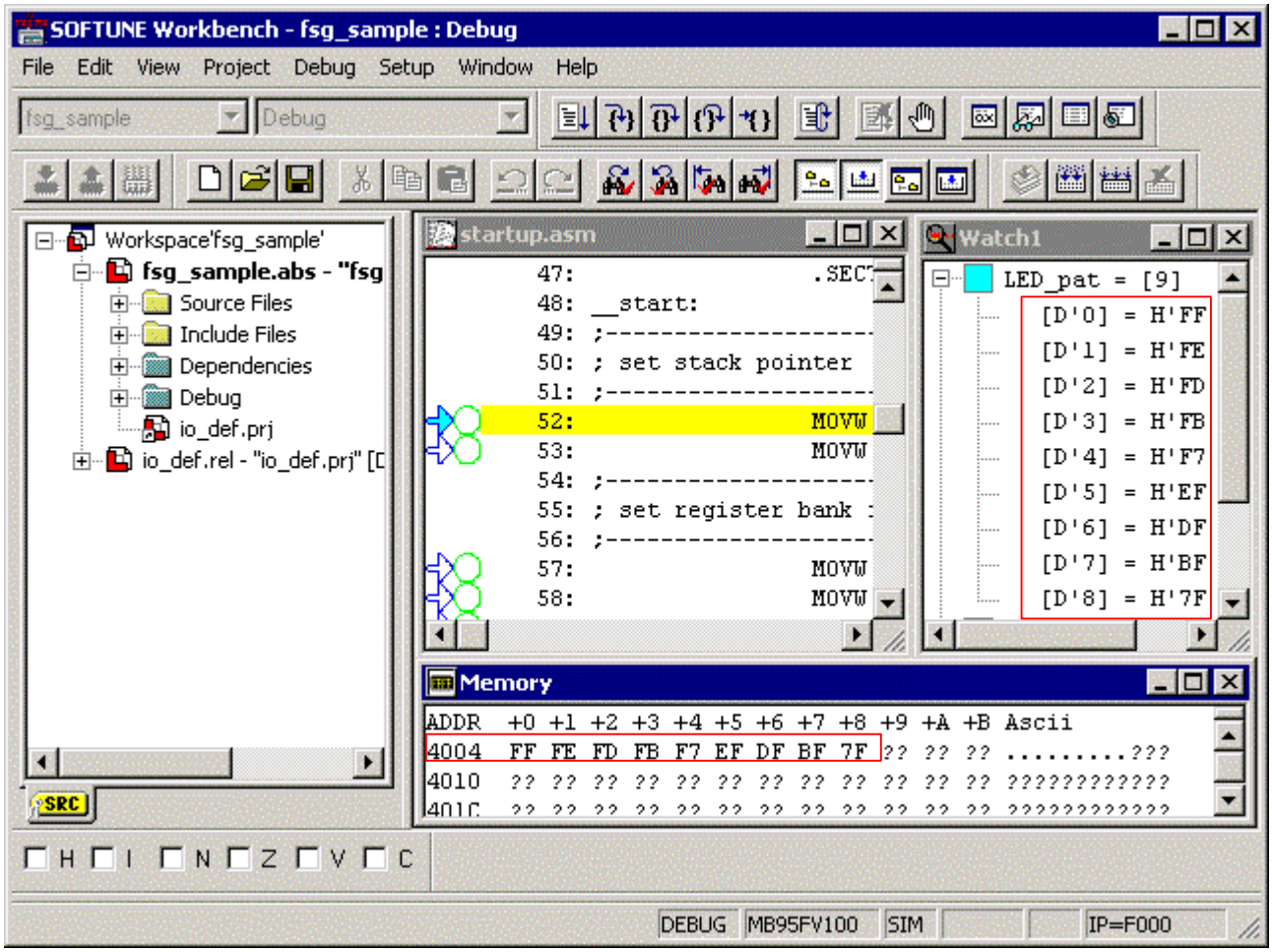
Similarly, register the following variables.



(3) Select [Memory] from the [View] menu to open the Memory window.
Specify the position in the [Jump] dialog to display address **LED_pat**.



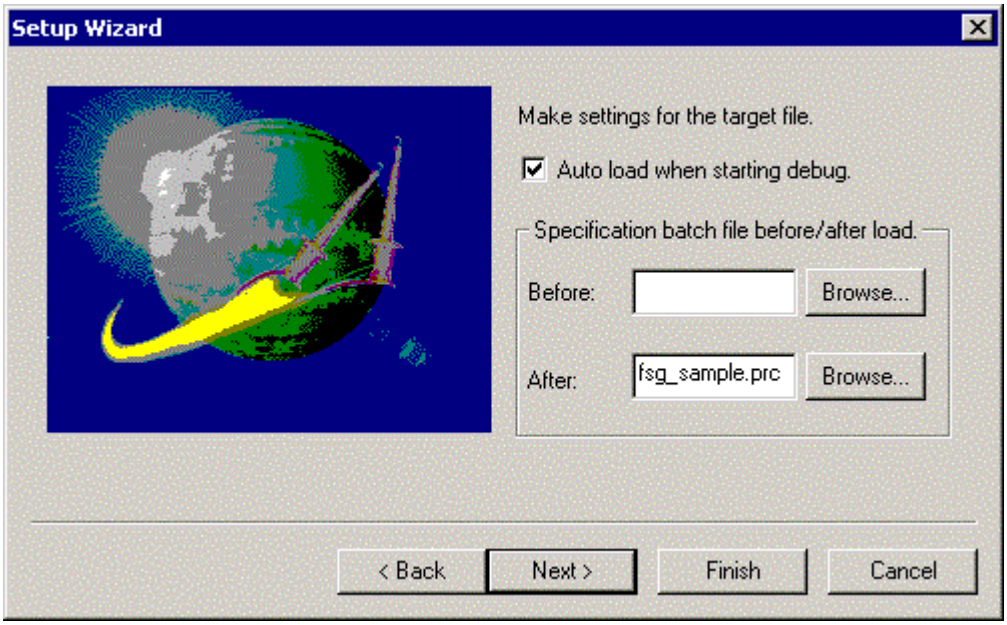
Check the description of the **CONST** variable (**LED_pat**).



One Point !

= Making batch file =

Making batch file enables execution of ‘Set interrupt’ automatically in switching to the debug mode.
By executing the batch file, the user can save in setting from the dialog.
The user can set automatic execution of the batch file in the Setup Wizard window.
Saving the file in the Project directory eliminates the need for specifying a path.



- Batch file sample **fsg_sample.prc**

```
if  %strcmp(%tostr(%DEBUGTYPE), "EML") ==0
endif
reset
cancel var /all

if  %strcmp(%tostr(%DEBUGTYPE), "SIM") ==0
        set interrupt /interval D'11,D'25000
endif
#

printf "\nThe preparations for the execution are completed. \n"
```

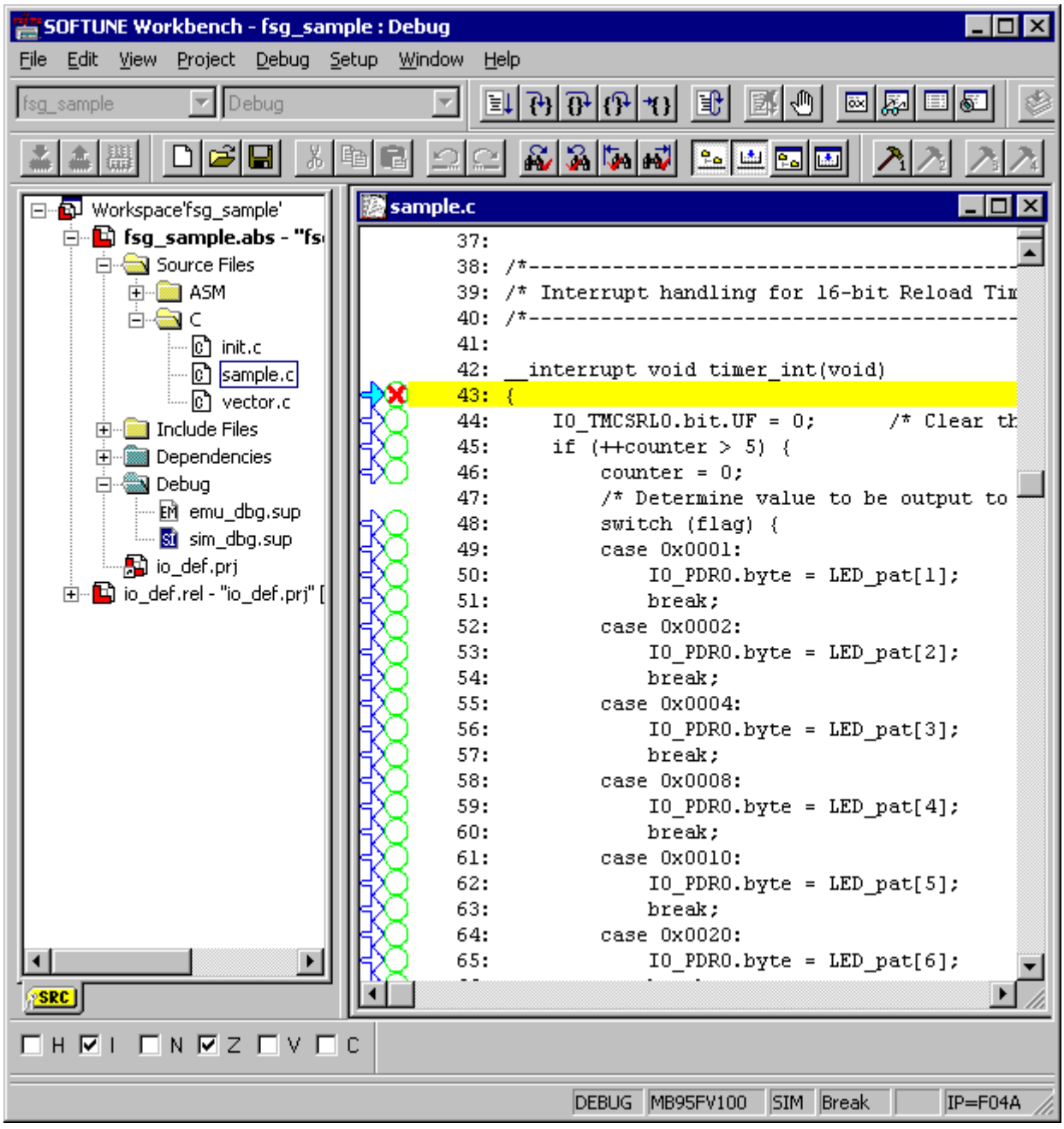
4.2.5 Setting Breakpoint

■ **How to Set Breakpoint**

During the Debugger executes a program, the execution of the program can be stopped when the program counter (PC) passes a certain address or accesses data at a certain address.
The position where the execution of the program is stopped is called a breakpoint. The user can set the code breakpoints and data breakpoints.
This sample sets a breakpoint at the beginning of the timer interrupt function in **`sample.c`**.

Click the left circle of the source window to set the breakpoint “X”.
The yellow line indicates the current position of the instruction pointer.

Note: The value of the instruction pointer(IP) is the same as that of the PC.



One Point !

= Types of breakpoints and how to set =

There are two types of breakpoint as follows:
Select [Breakpoints] from the [Debug] menu to open the [Break] dialog.
This dialog can also set breakpoints. It also provides a list of set breakpoints.

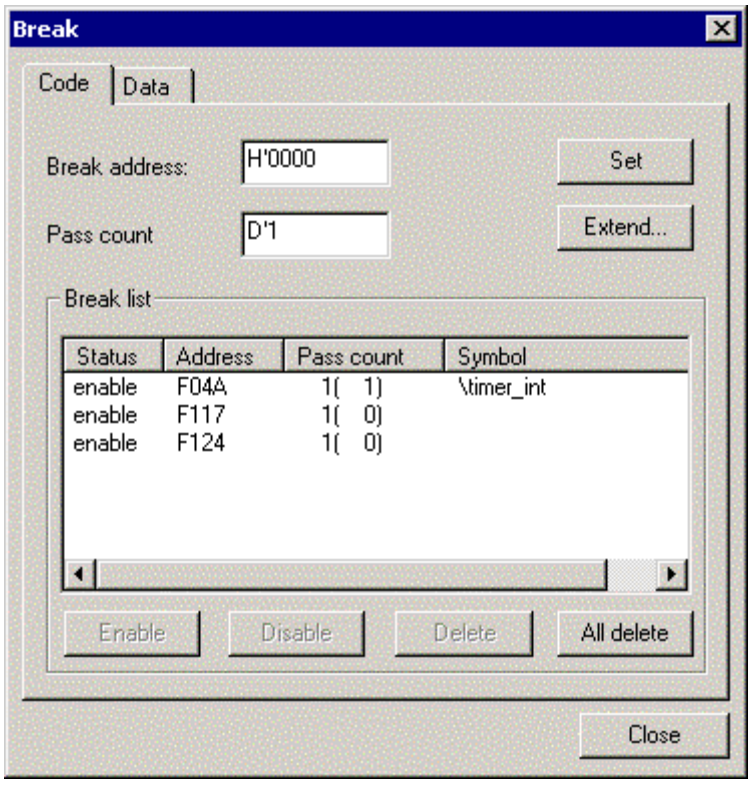
- Code breakpoint

  Code breakpoint is a breakpoint to stop the program when the PC attempts to execute a set address. For both Simulator and Emulator, the PC stop position is a breakpoint before execution.

- Data breakpoint

  Data breakpoint is a breakpoint to stop program when the PC accesses data at a set address.

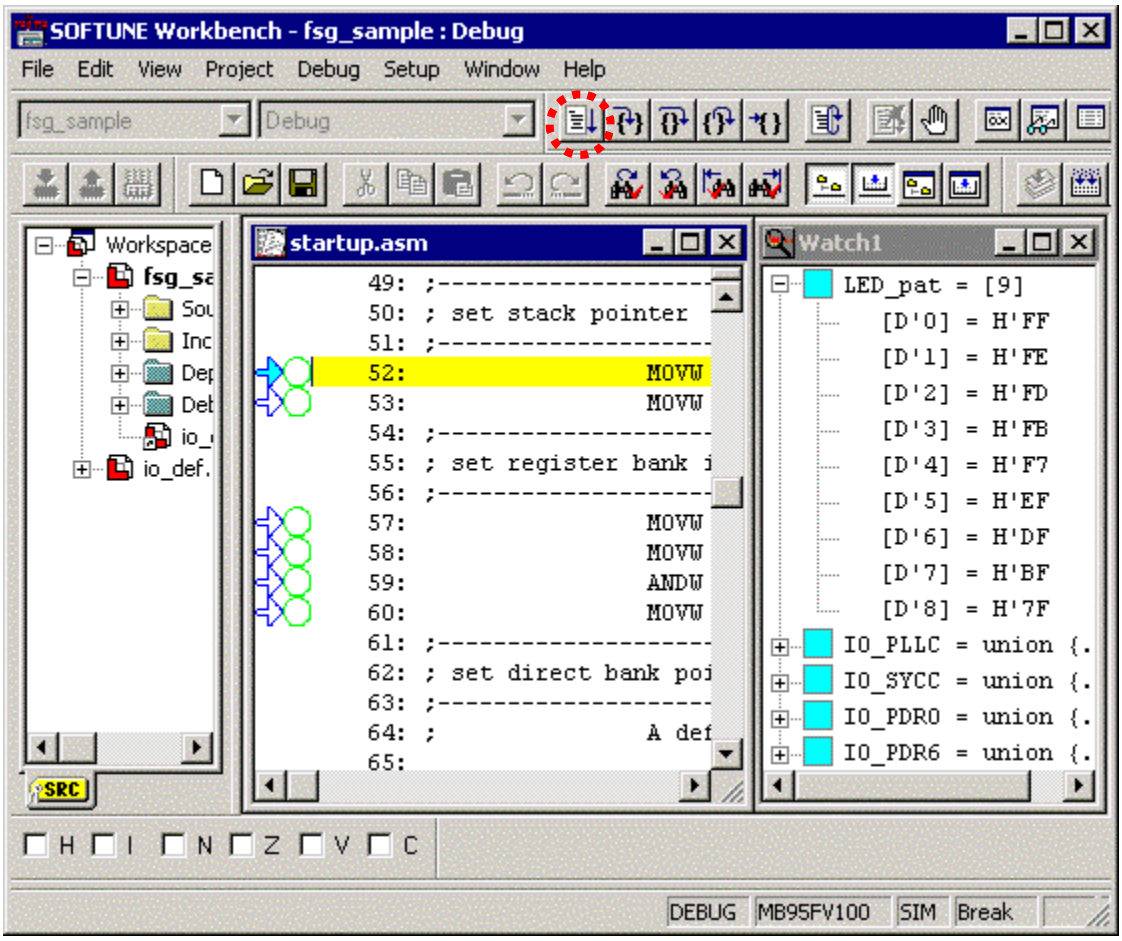
  Read, write, and read/write can be set as access conditions.



4.2.6 Executing and Stopping Program

The program is executed up to the breakpoint set in ***Section 4.2.5***.
This sample includes the program for clock switching. However, because the Simulator has no resources, the PLL clock enters the oscillation stabilization wait state.
Stop executing the program temporarily and set “1” to the MPRDY bit of the PLL control register (IO_PLLC) in the Watch window, and then set “1” to the SCM0/SCM1 bit of the system clock control register (IO_SYCC).
When the user reruns the program, the program stops at the set breakpoint.

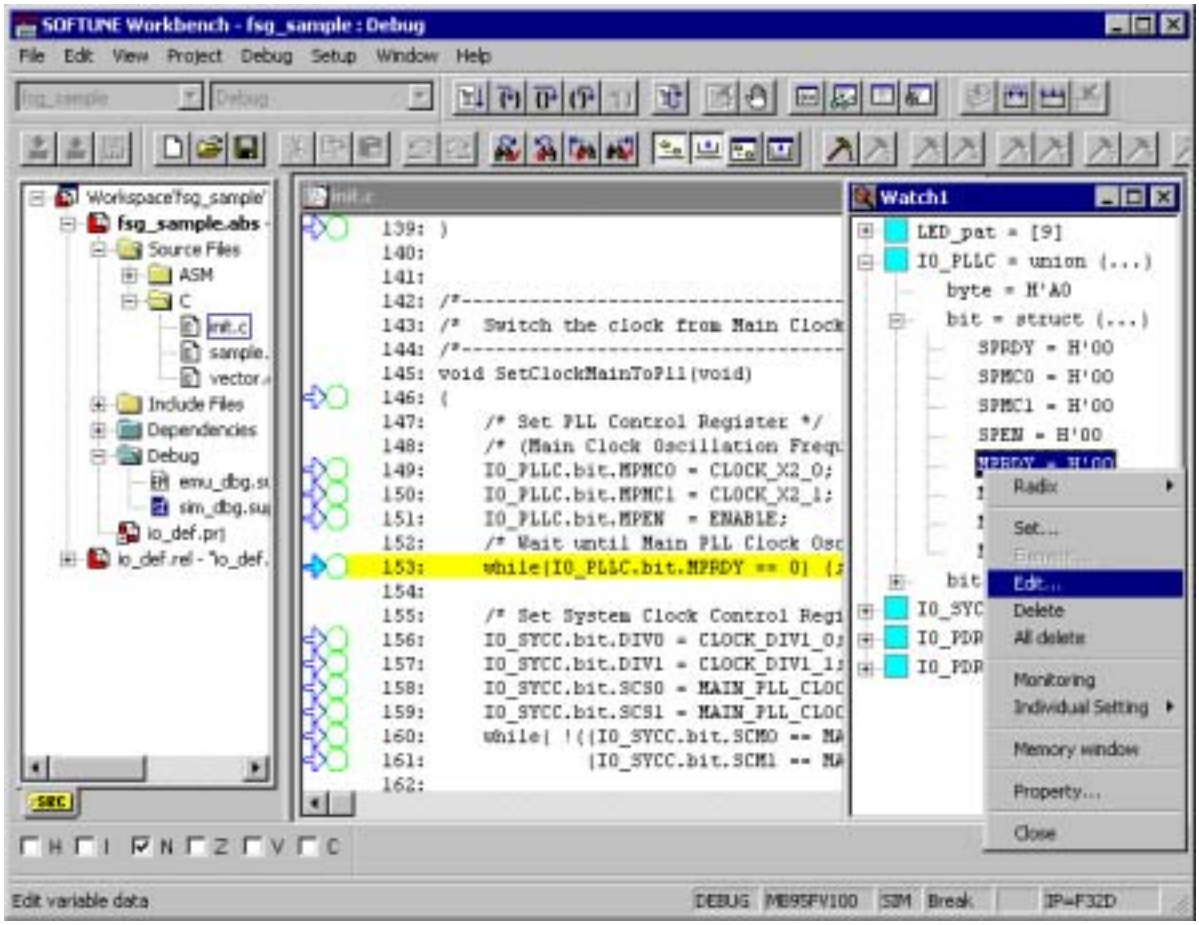
(1) Click the [Run continuously] button (enclosed in the dotted lines) to execute the program.

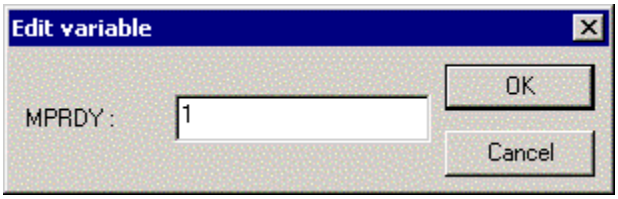


(2) Click the [Stop execution] button (enclosed in the dotted lines) to stop the program temporarily.

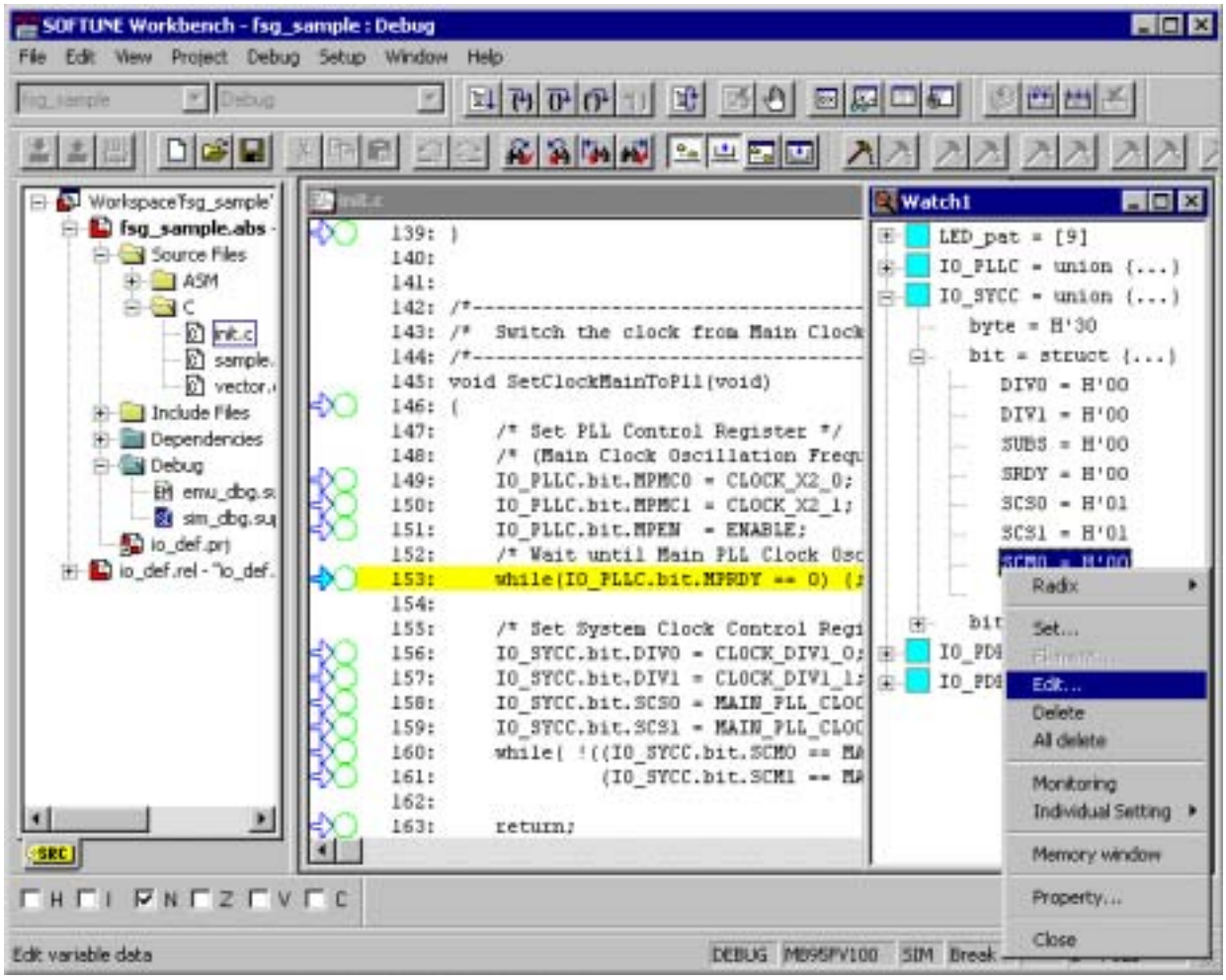
(3) Edit the MPRDY bit of the **IO_PLLC** register in the Watch window.

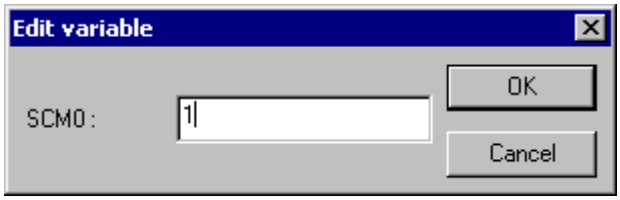
Select the MPRDY bit and [Edit...] from the shortcut menu, and then set “1” to the MPRDY bit in the [Edit variable] dialog.

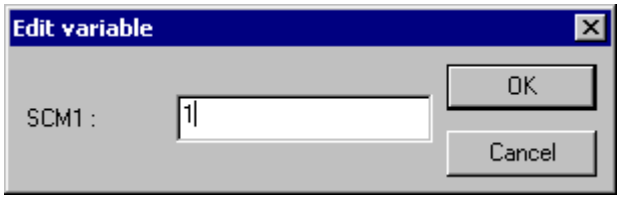




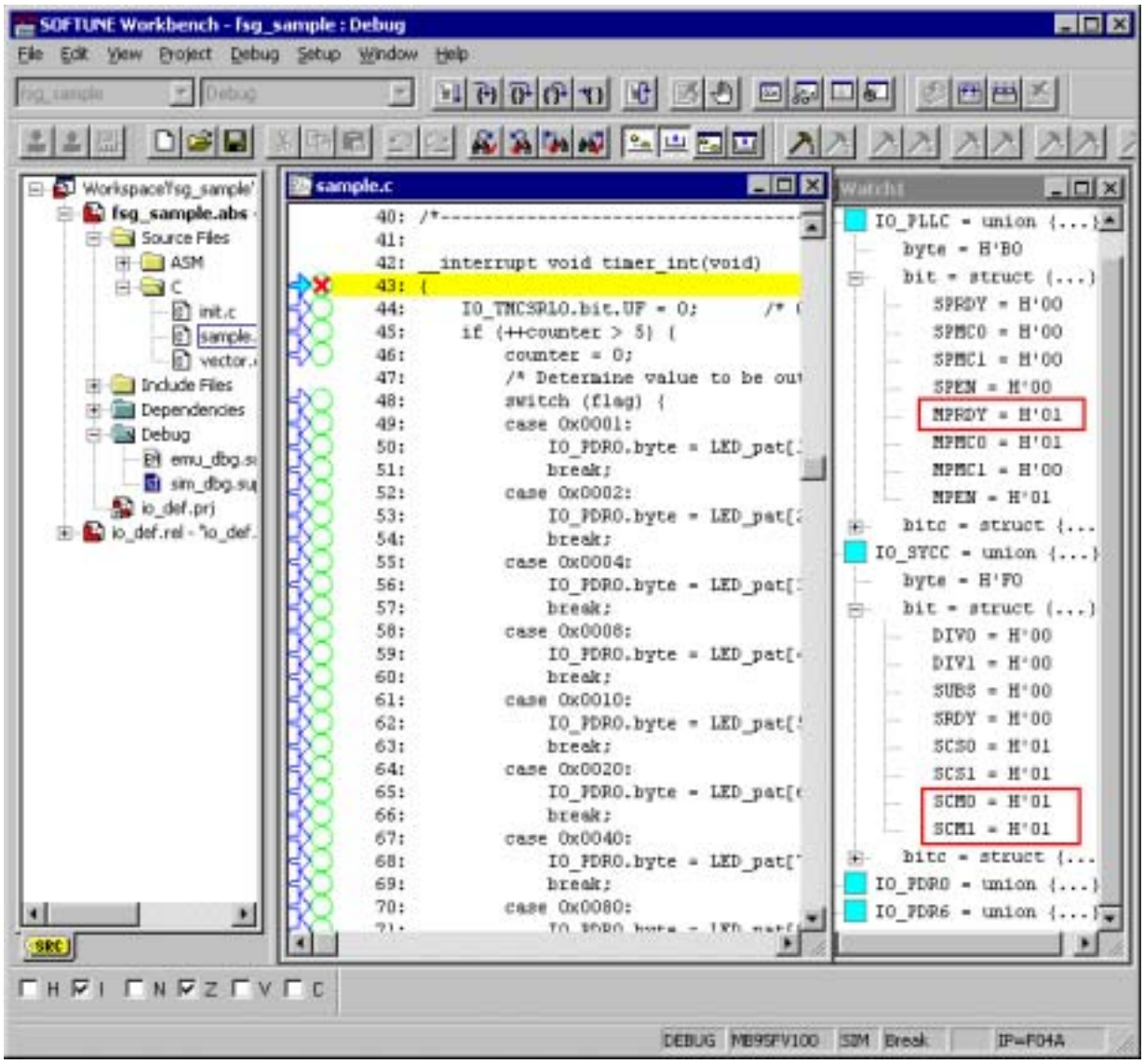
(4) Edit the SMC0 and SCM1 bit of the IO_SYCC register in the Watch window.
Select the SMC0/SMC1 bit and select [Edit...] from the shortcut menu, and then set “1” to the SMC0/SMC1 bit in the [Edit variable] dialog.







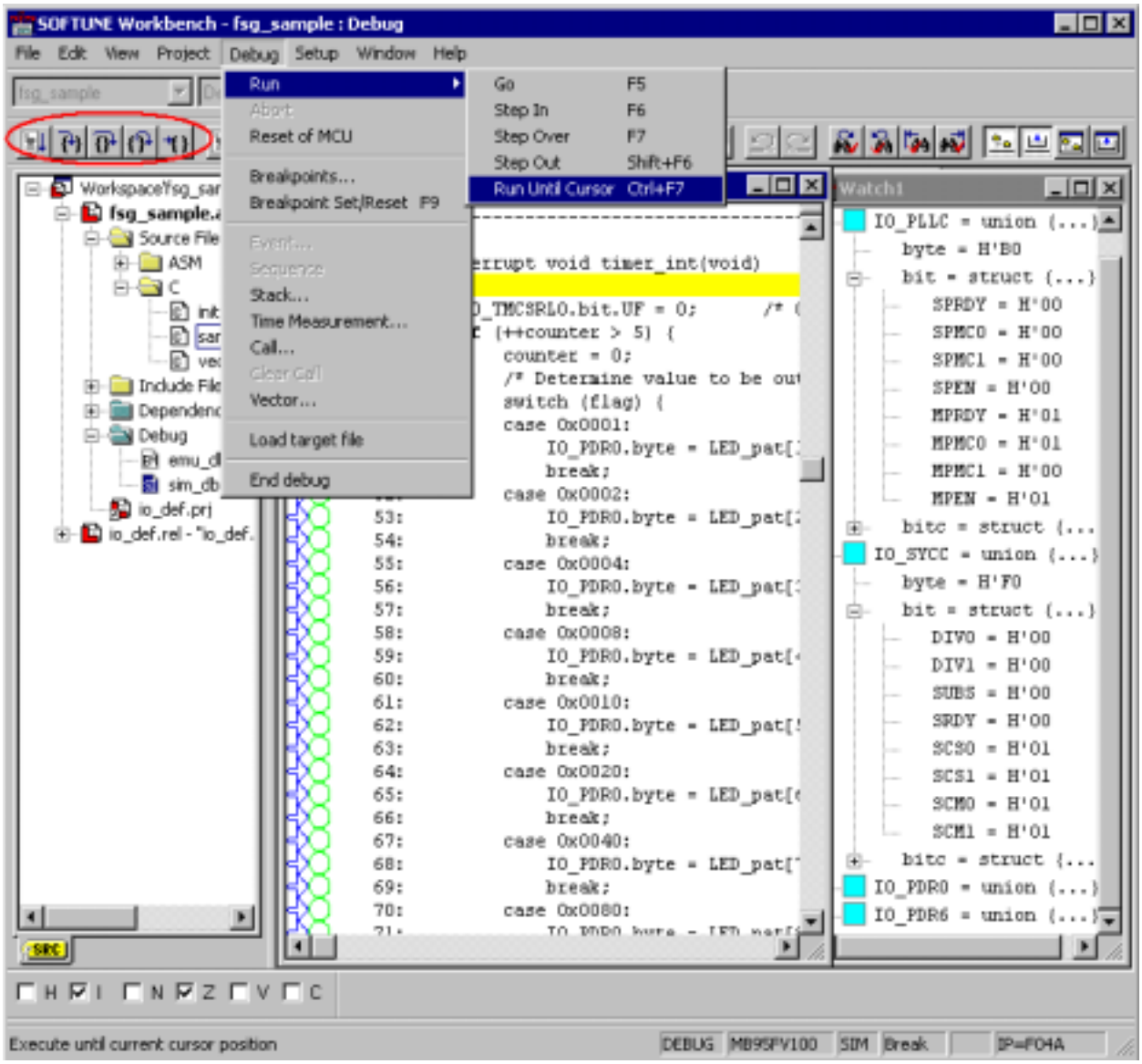
(5) When the user reruns program, the program is executed up to the breakpoint.



One Point !

= Program execution by Debugger =

Use the [Run] command in the [Debug] menu to execute program using the Debugger.
The use of the buttons enclosed in line also executes program.



(1) Run continuously

The program is executed continuously from the current PC position.

The program being executed stops when the program reaches the breakpoint or [Abort] is selected from the [Debug] menu.

(2) Step In

Step-by-step program execution is performed and stopped after moving the PC to the address of the next instruction.

When the function call instruction is executed, the program execution is stopped at the beginning of the function.

(3) Step Over

Step-by-step program execution is performed and stopped after moving the PC to the beginning of the next instruction.

When the function call instruction is executed, all the functions are executed and the program execution is stopped after moving the PC to the address of the instruction next to the function call instruction.

(4) Step Out

The current function is executed up to its end, returning to the function call source and program execution is stopped after moving the PC to the address of the instruction next to the function call instruction.

(5) Run Until Cursor

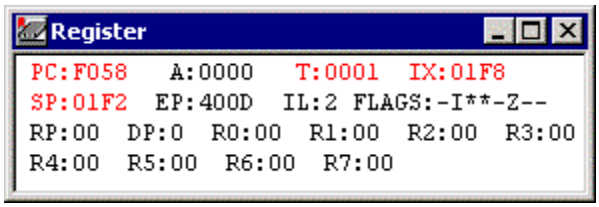
Program execution is performed up to the immediately preceding instruction at the address at which the current cursor is on the Source or Disassemble window, and stopped after moving the PC to the address at the cursor.

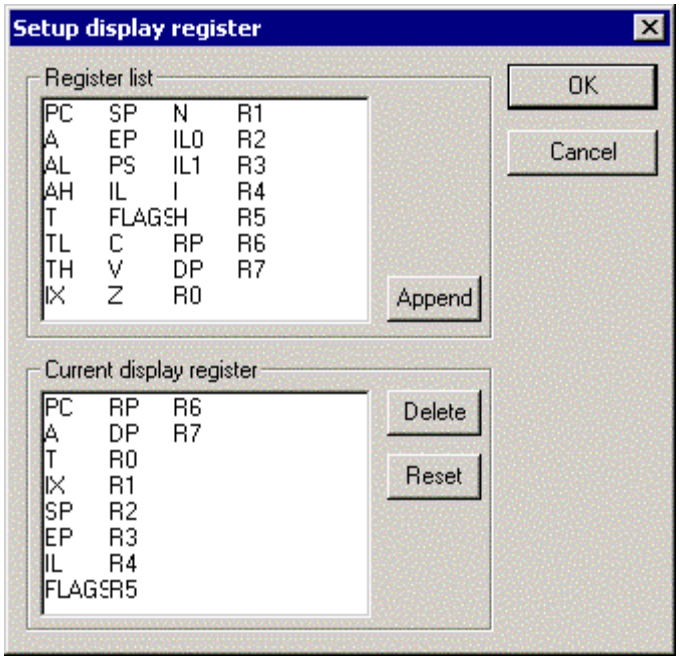
To reset the program execution, click [Reset of MCU] in the [Debug] menu or the button enclosed in dotted line.

One Point !

= Displaying Register window =

When [Register] is selected from the [View] menu, the Register window is opened.
The register with a value changed by executing the program is displayed in red on the Register window.
To display the register, select [Setup] from the shortcut menu. Then, select the register to be displayed in the [Setup display register] dialog.

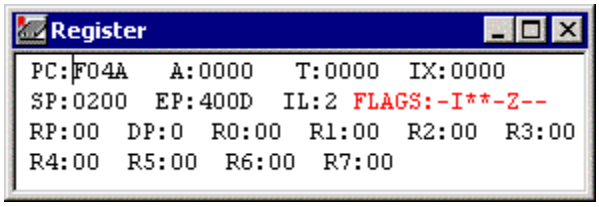




One Point !

= Status display of MCU condition flag =

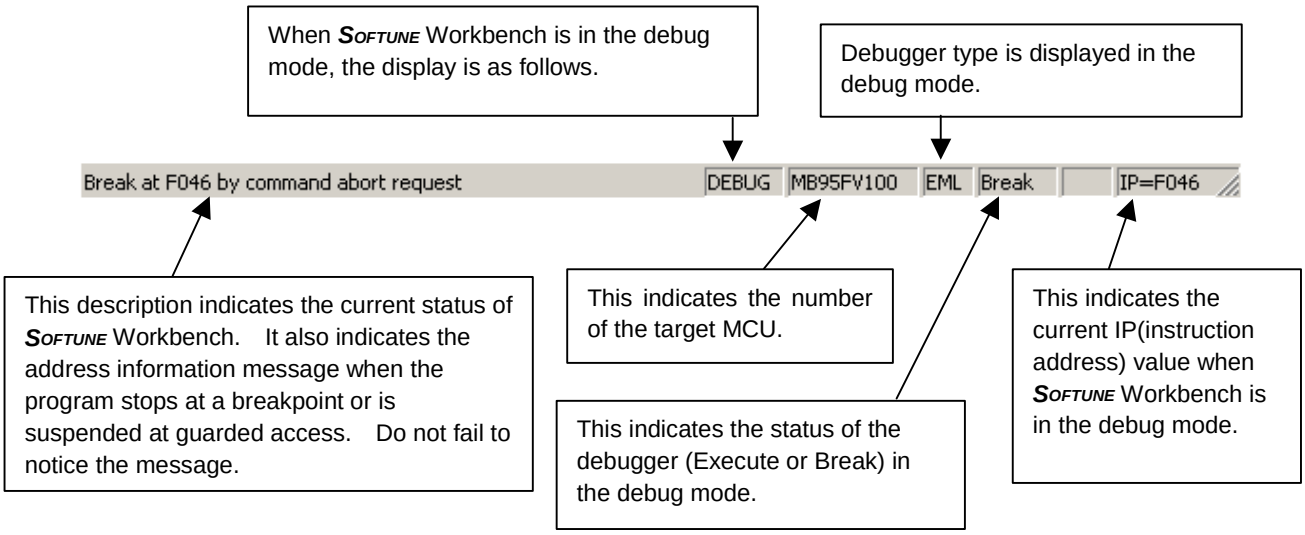
The status of the condition flag is displayed.
Clicking this checkbox enables to reset or clear each flag.
If the checkbox is not displayed, select [Flag] from [Tool bar] in the [View] menu.



One Point !

= Displaying status bar =

The status bar displays the following descriptions.
If the status bar is not displayed, select [Status Bar] from the [View] menu.



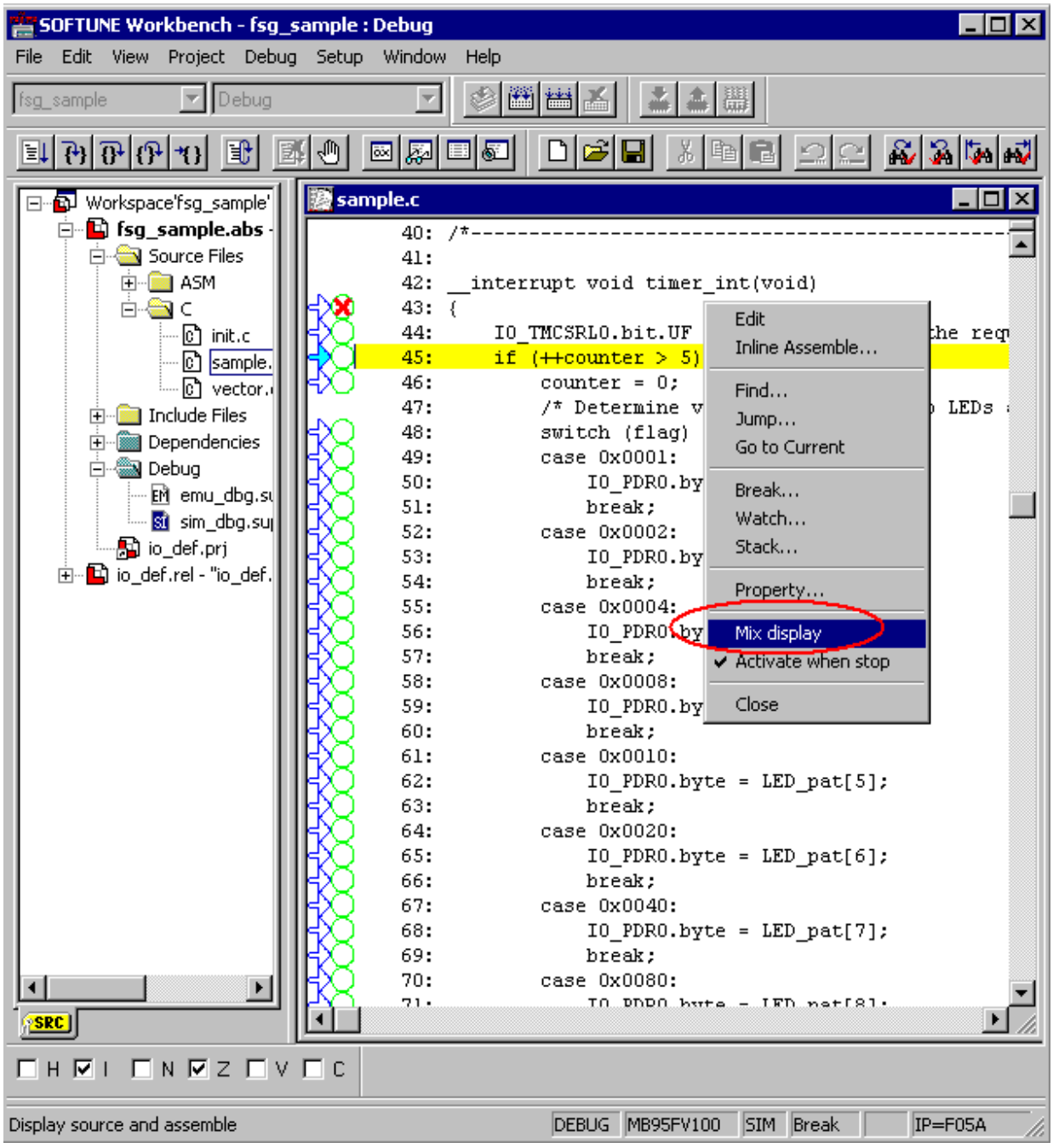
4.2.7 Mix Display

■ **Mix display**

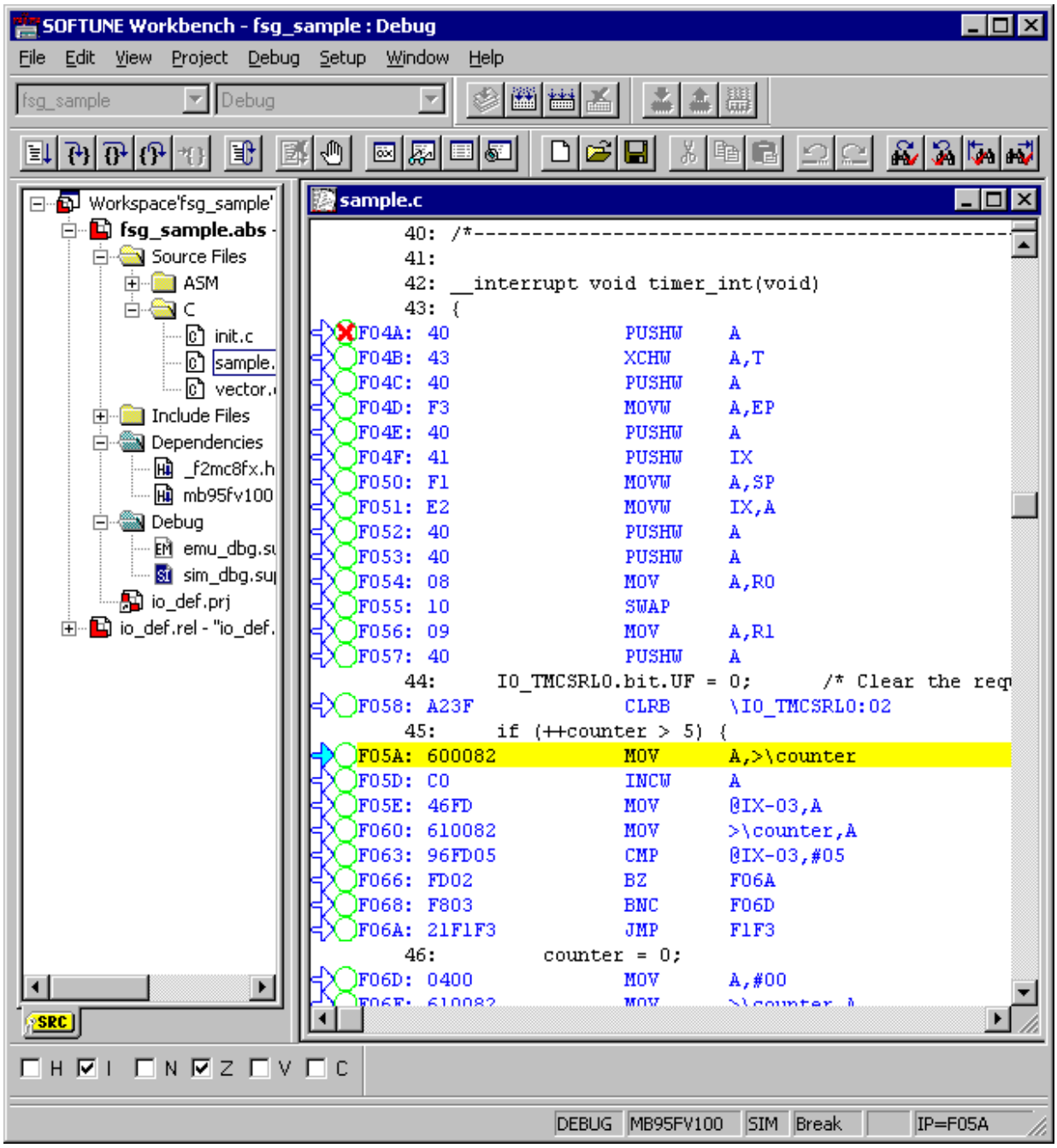
When [Mix display] is selected during debugging, the C source program and disassembled code are displayed at the same time, making step-by-step program execution possible at disassemble level.

《 Setting procedure 》

(1) Select [Mix display] from the shortcut menu in the Source window (for displaying the source program at debugging).



(2) The source program and disassembled code are displayed at the same time.



4.2.8 Monitoring

■ **Monitoring**

The [Memory window] and [Watch window] provide the monitoring function.
Using the monitoring function, the window value is updated on as needed basis even during executing the program.
Select the window to be monitored and set the sampling time in the [Setup debug environment] dialog.

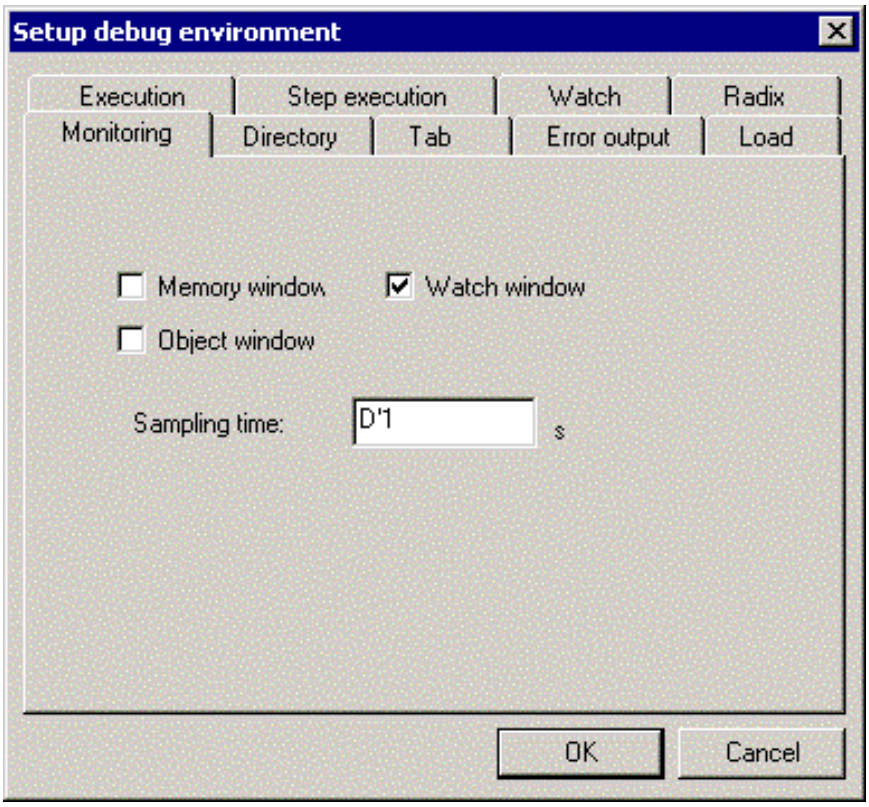
《 Setting procedure 》

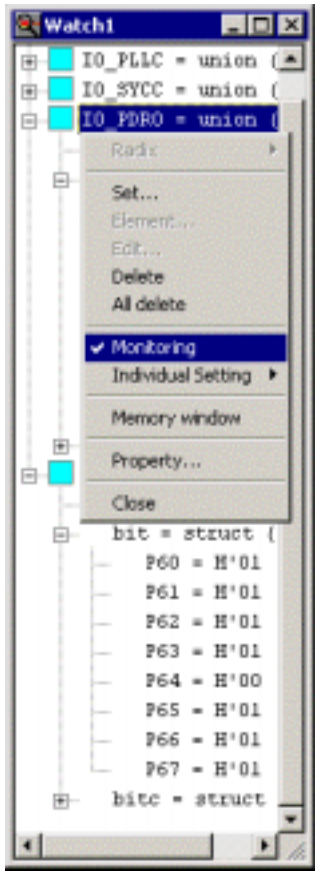
(1) Select [Debug Environment…] from [Debug Environment] in the [Setup] menu and open the [Setup debug environment] dialog.

Select the [Monitoring] tab and mark the [Watch window] checkbox.

The shortcut menu in the Watch window also enables selection of [Monitoring].

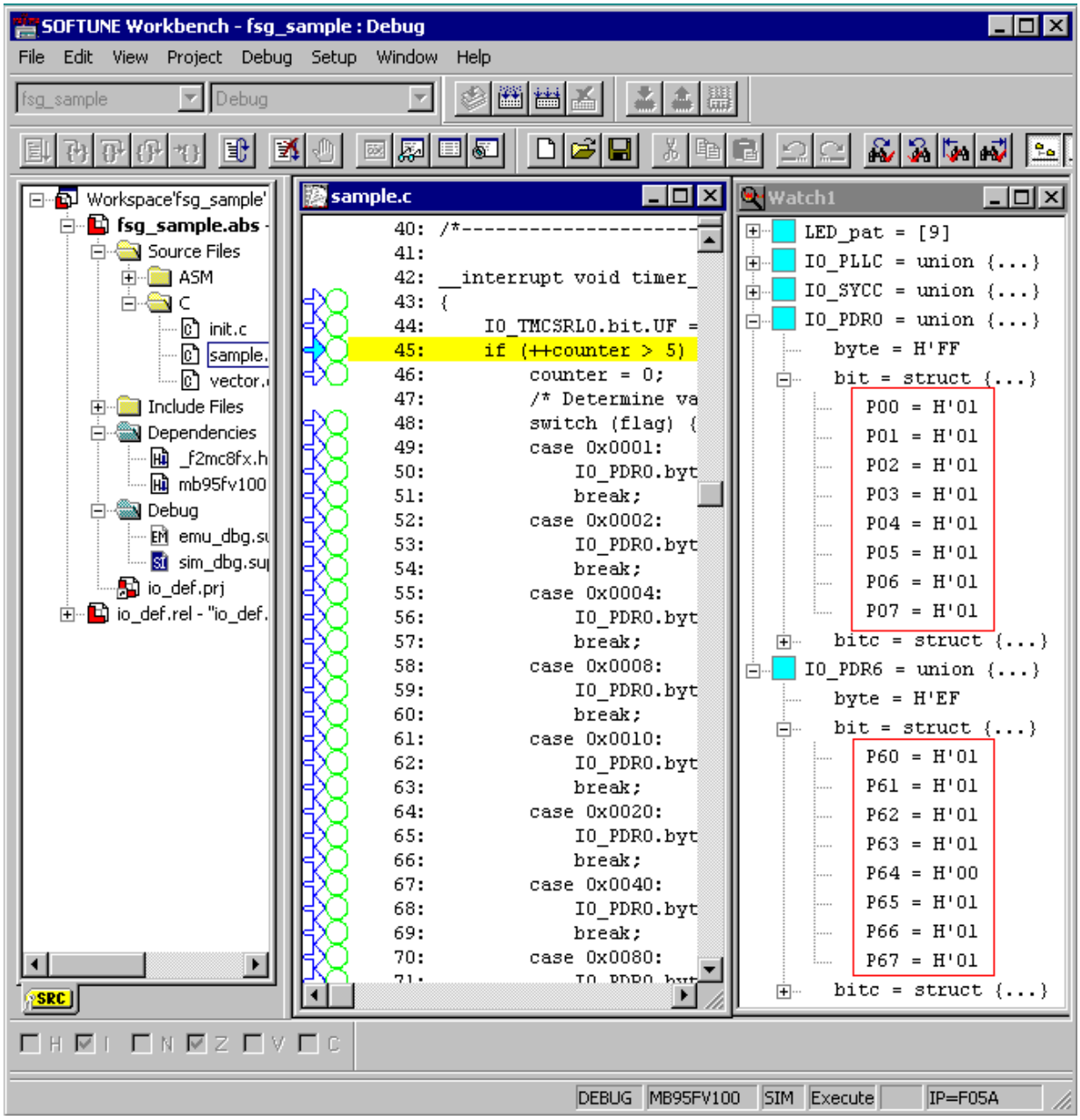
The sampling time can be set in 1-second units.





(2) Click the breakpoint to deselect it.

The variables in the Watch window are always updated when the program is executed.



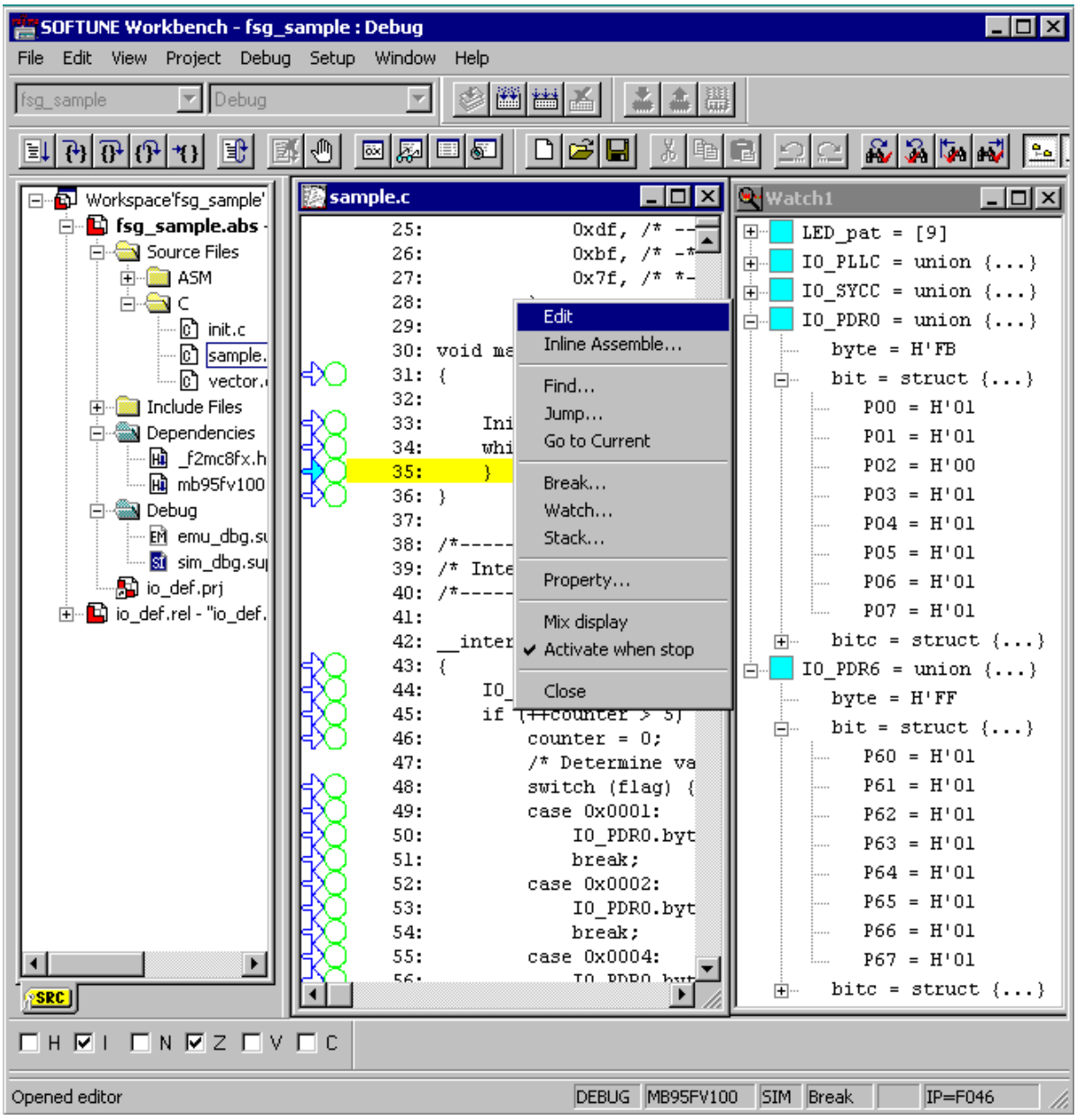
4.2.9 Correcting and Re-debugging Program

■ Correcting and Re-debugging Program

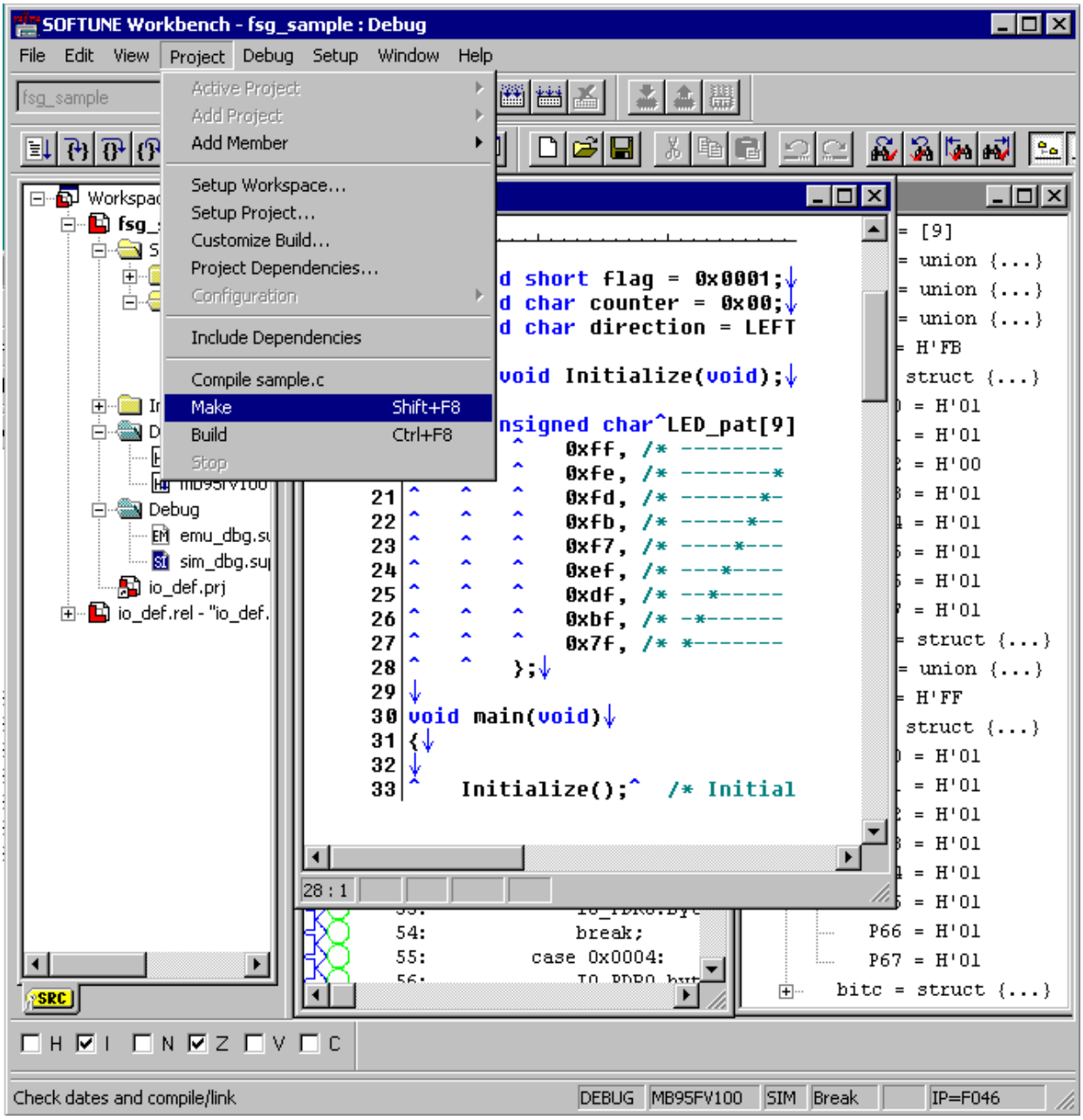
This section describes the procedure for correcting and re-debugging the source file.
The user can remake and re-debug the target file without completing debugging.
This procedure can be also used for the Emulator Debugger.

《 Setting procedure 》

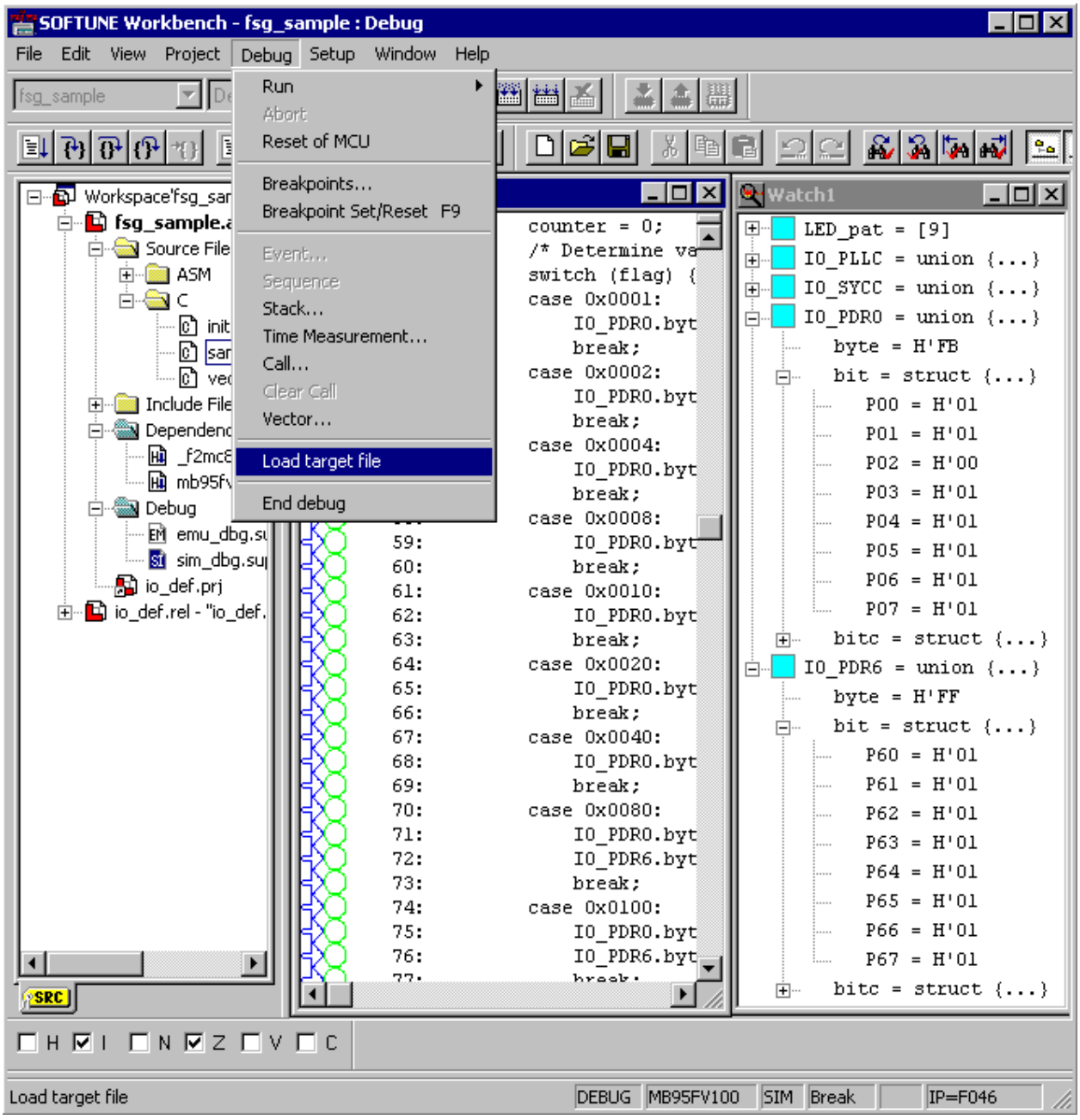
(1) Right-click the Debugger Source window to select the [Edit] command from the shortcut menu.
When the Edit window for the source file appears, correct the source file.



(2) Execute Make to make a target file.



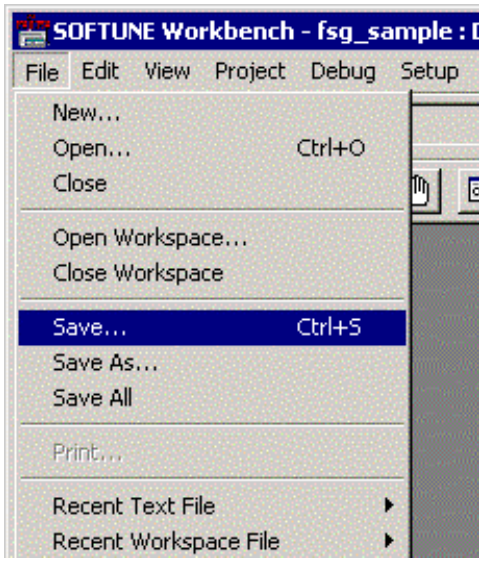
(3) Execute [Load target file] in the [Debug] menu to load the remade target file. In this sample program, this step may be omitted because [Auto load the target file after Make/Build] is set in [Setup Workspace].

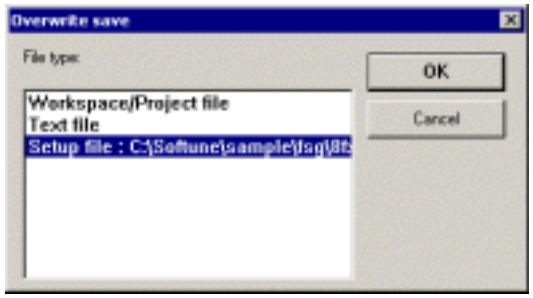


One Point !

= Saving setup file for Simulator =

The setup file is not saved automatically until debugging is terminated. When updated, save the setup file. Select the [Save] command from the [File] menu and save the setup file for Simulator in the [Overwrite save] dialog.
Saving the setup file is valid only in the debug mode.





4.3 Debugging by Emulator

■ Debugging by Emulator

This section describes the procedure for actual machine verification using the Emulator.
The equipment used for explanation of the Emulator is listed below:

- MB2146-09 : BGM Adapter
- MB2146-30X : MCU Board (Super EVA containing MB95V100-10X)
- MB2146-401 : Standard evaluation board for F^2MC-8X
- MB95FV100 : Evaluation chip
- Crystal oscillator : 4 MHz
- Power supply for MB2146-401 : 5 VDC
- USB(A-B) cable : 1.1USB

<MB2146-401 Settings>

- JP 3,4 : Switches 3 and 2 shorted (AC adapter: +5.0 V)
- JP 1,2,5,6,7 : Jumper fitted

■ Emulator Functions

- 4.3.1 Loading Monitor Program
- 4.3.2 Making Setup File by Setup Wizard for Emulator
- 4.3.3 Executing Program
- 4.3.4 Setting Breakpoint
- 4.3.5 Trace

4.3.1 Loading Monitor Program

■ **Download Monitor Program**

Using the Emulator Debugger requires downloading the monitor program to the ICE according to the chip to be used. This ICE checks the type and version of the monitor program at the start of debugging and loads the monitor program automatically.

This sample gives an example of the connection between the ICE (MB2146-09) and personal computer via the USB.
For details, refer to the following manuals:

- ***Workbench Operation Manual Appendix B Downloading Monitor Program***
- ***Workbench Operation Manual Appendix D Setting USB Interface***

(1) Connect the ICE to a personal computer via the USB.

For details of how to connect, refer to the manual attached to the ICE.

4.3.2 Making Setup File by Setup Wizard for Emulator

■ **Procedure for Setting Setup File by Setup Wizard**

This section explains the procedure for making an additional setup file for the Emulator incase where the setup file for the Simulator already is made.

《 Setting procedure 》

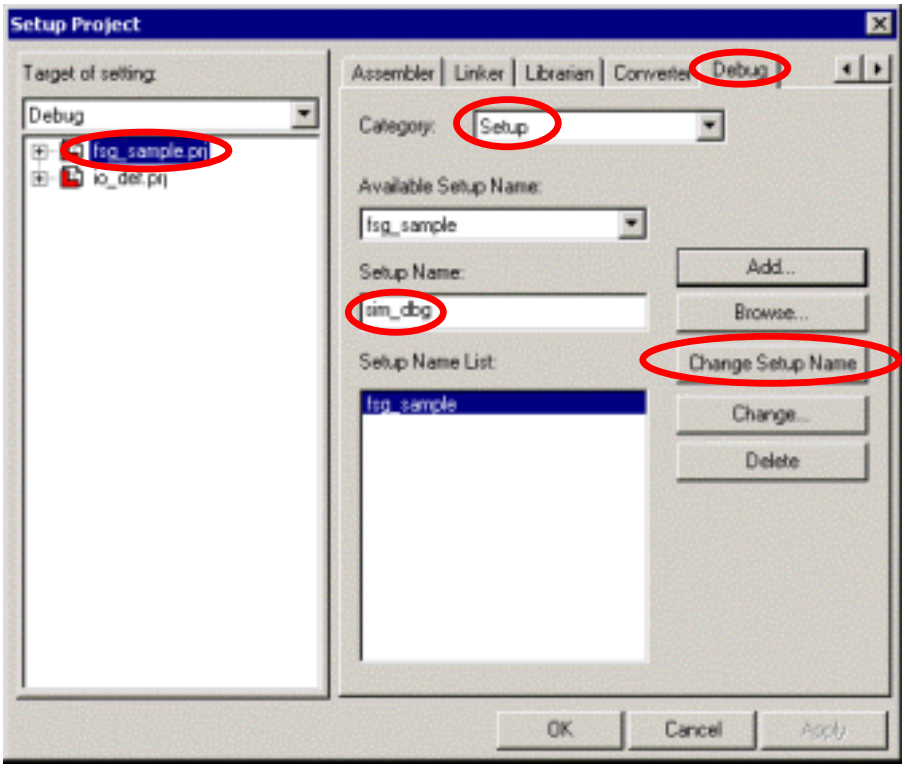
(1) Select [Setup Project] from the [Project] menu.

A setup file cannot be made in the debug mode. Use the [Terminate debug] command in the [Debug] menu to terminate the debug mode.

(2) Change the setup name for the Simulator Debugger.

Click "**`fsg_sample.prj`**" in "Target of setting" and select "Setup" at "Category" in the "Debug" tab.

Change "**`Setup name`**" to "**`sim_dbg`**" and click the [Change Setup Name] button.

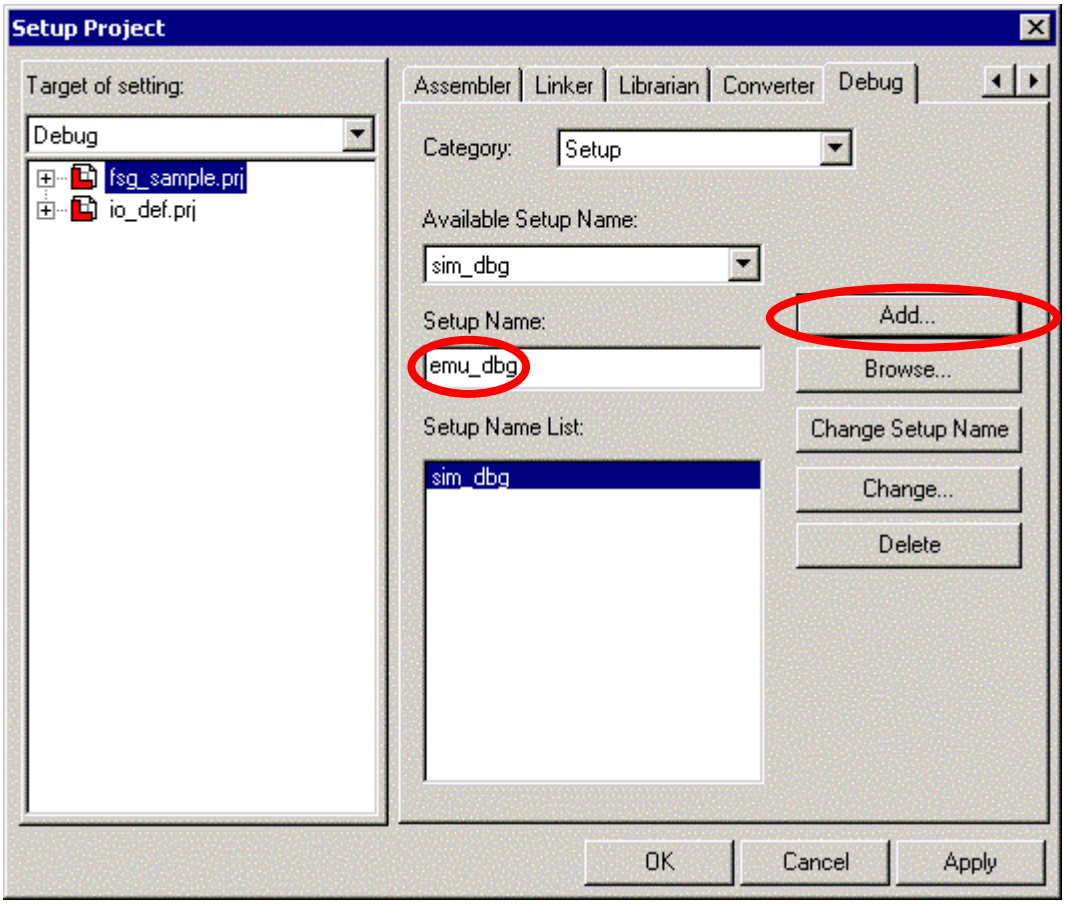


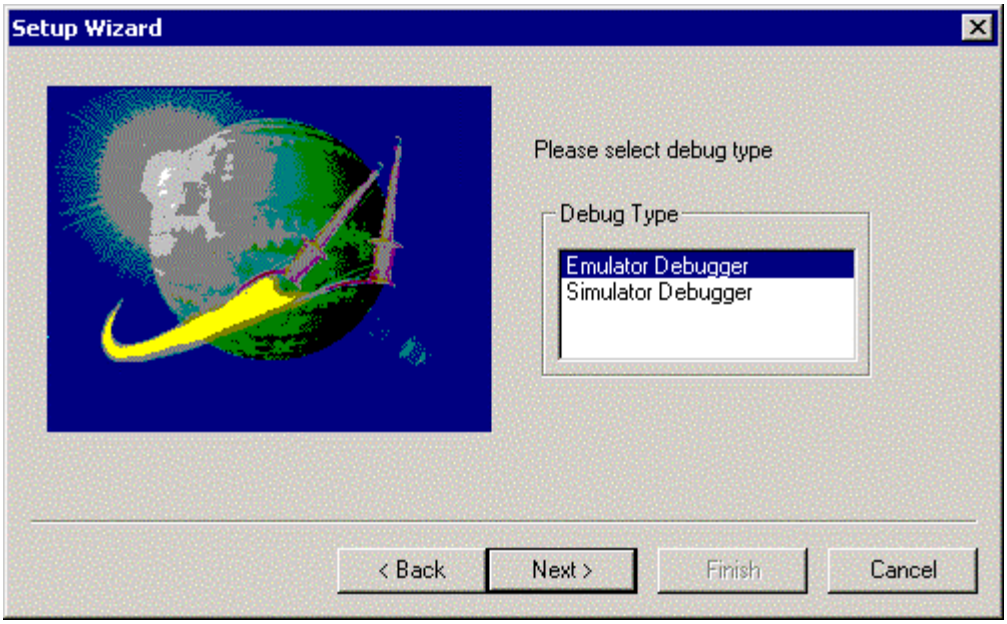
(3) "**sim_dbg**" is added as the new setup name to "Setup Name List."

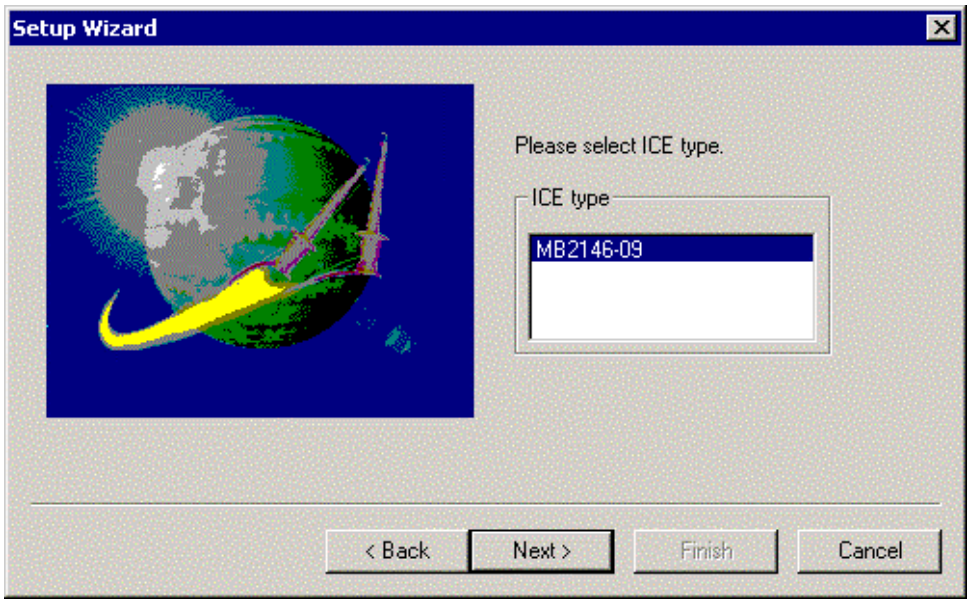
Make a setup file for the Emulator.

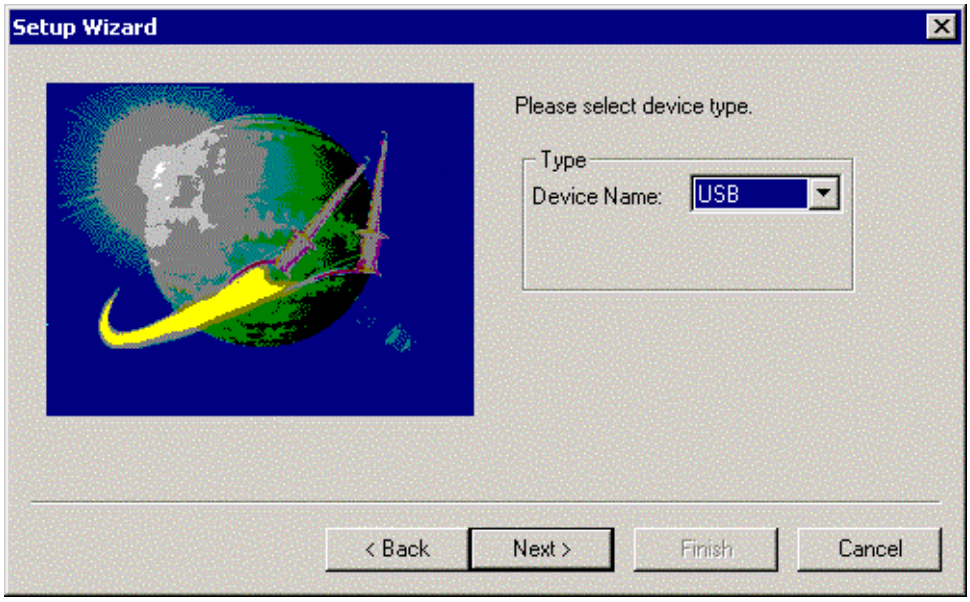
Enter "**emu_dbg**" for "Setup Name" and click the [Add...] button.

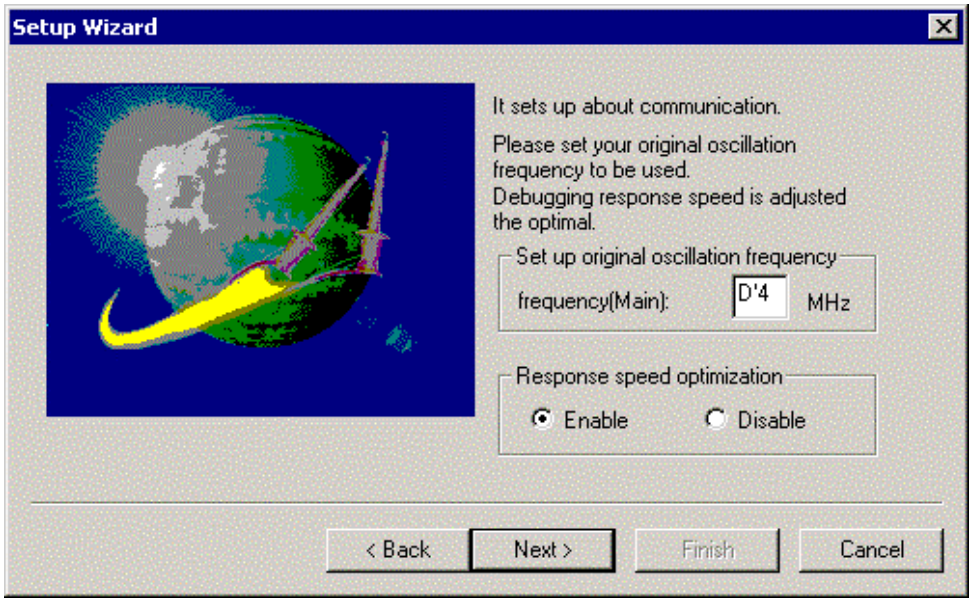
After the setup wizard starts up, set according to the menu.

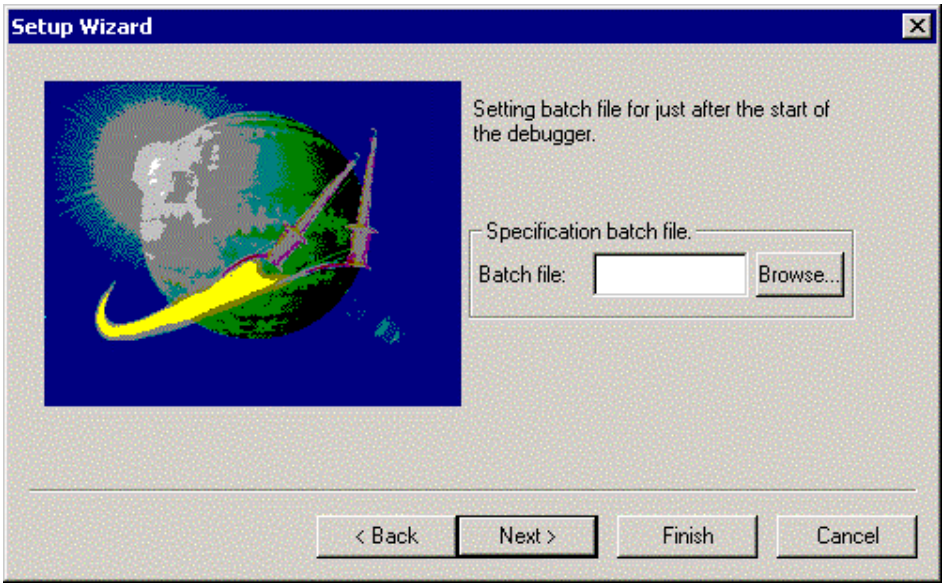


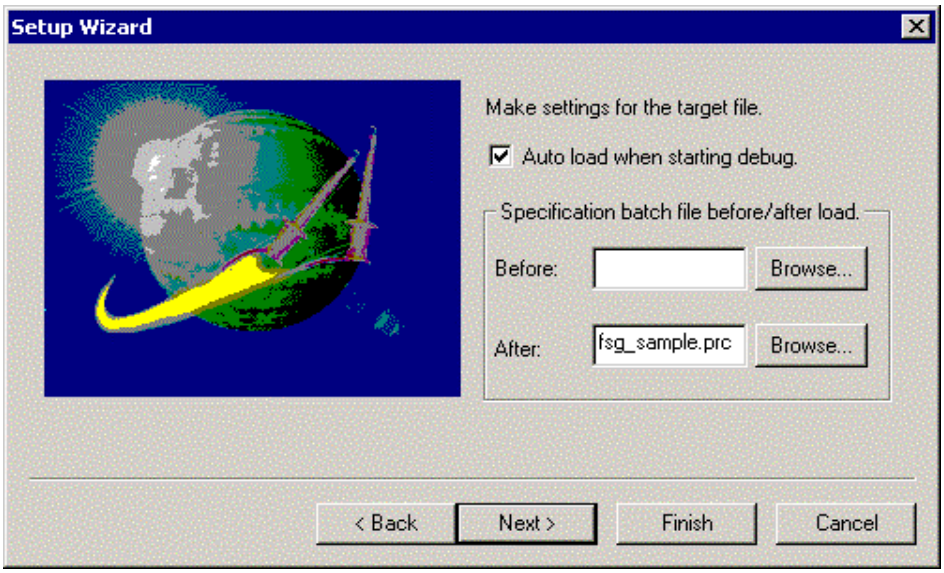


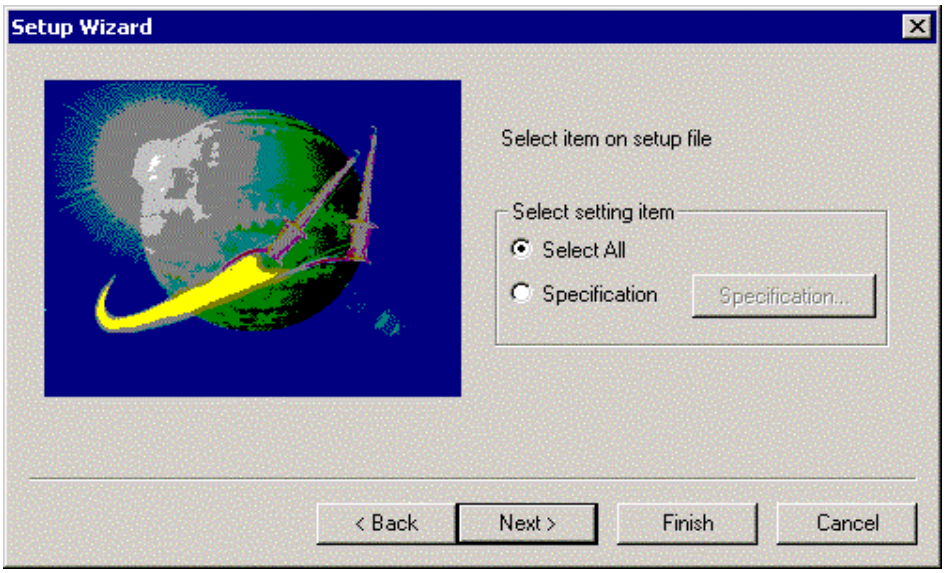

Setup Wizard
Please select ICE type.
ICE type
MB2146-09
< Back
Next >
Finish
Cancel

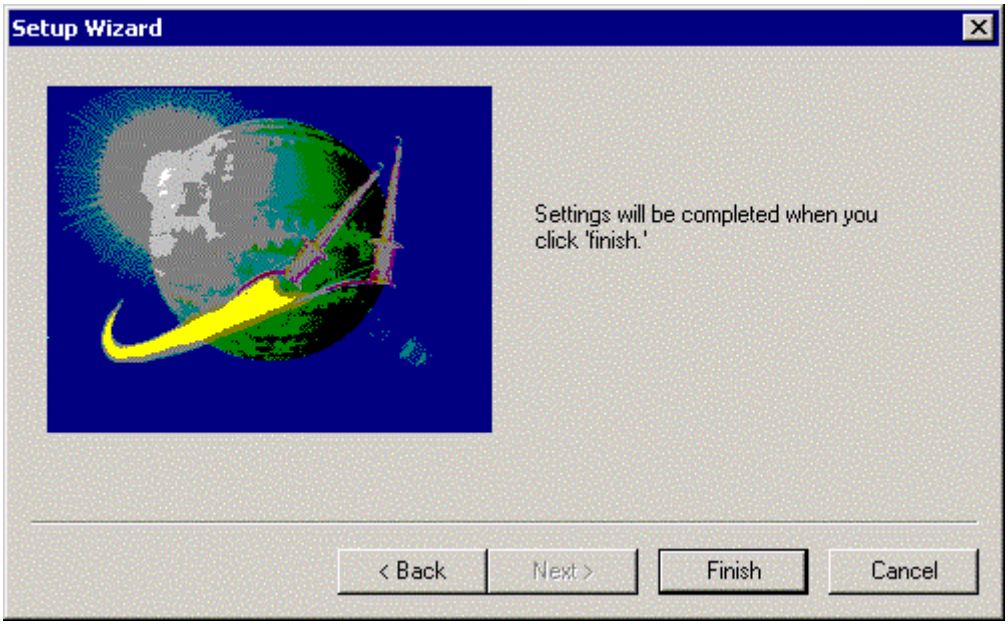

Setup Wizard
Please select device type.
Type
Device Name:
USB
< Back
Next >
Finish
Cancel


Setup Wizard
It sets up about communication.
Please set your original oscillation frequency to be used.
Debugging response speed is adjusted the optimal.
Set up original oscillation frequency
frequency(Main):
D'4
MHz
Response speed optimization
Enable
Disable
< Back
Next >
Finish
Cancel

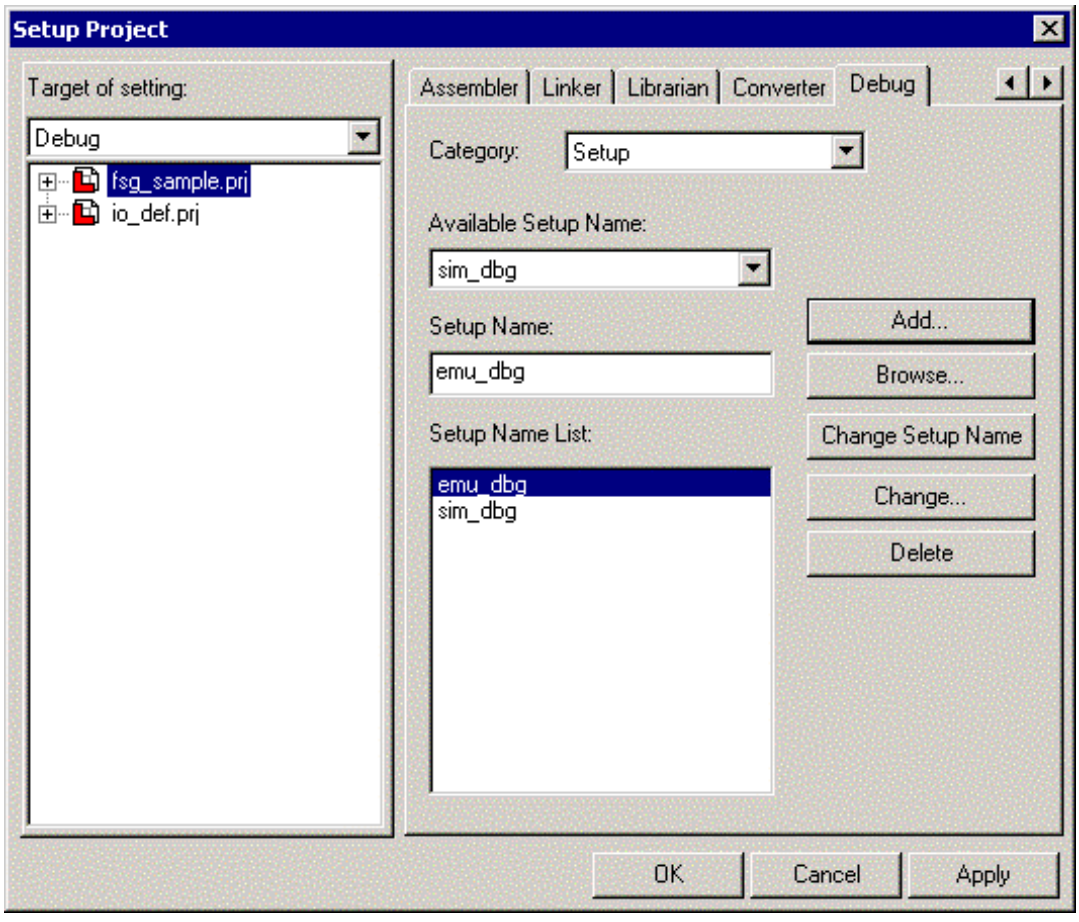

Setup Wizard
Setting batch file for just after the start of the debugger.
Specification batch file.
Batch file:
Browse...
< Back
Next >
Finish
Cancel


Setup Wizard
Make settings for the target file.
Auto load when starting debug.
Specification batch file before/after load.
Before:
Browse...
After:
fsg_sample.prc
Browse...
< Back
Next >
Finish
Cancel


Setup Wizard
Select item on setup file
Select setting item
Select All
Specification
Specification...
< Back
Next >
Finish
Cancel



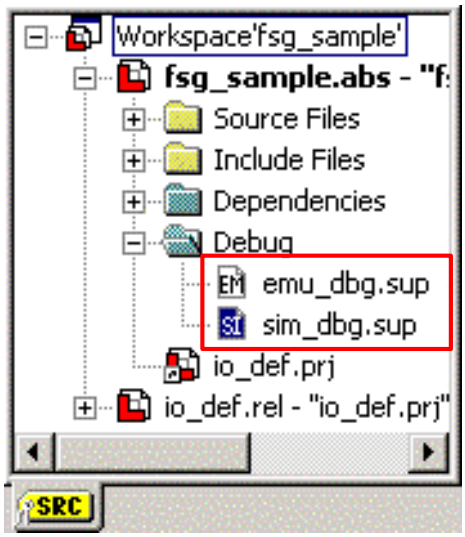
(4) After completing the settings in the Setup Wizard window, click the [OK] button. The setup file for the Emulator is added.



(5) The added setup file appearsappears on the window.

The setup file, which starts up when the [Start debug] command in the [Debug] menu is executed, is displayed in blue.

Double-clicking the setup file, the Debugger starts up according to its setup information.



One Point !

= Optimization of response speed during debugging =

Optimization of response speed optimization during debugging, which is one of the settings related to communications, determines whether to change the CPU clock to increase a response speed during debugging when the user program breaks. Besure to return the CPU clock to its original value when executing the user program. That means when this setting is enabled, the CPU clock overclocks to increase the communication speed between the host and the target, and the response speed during debugging (such updating time of flash memory) only when the user program breaks. When this setting is disabled, contrary the CPU clock remains unchanged even when the user program breaks and the communication speed between the host and the target depends on the CPU clock speed set by the user.

For details about how to change the setting of response speed optimization during debugging, see ***APPENDIX B Q13 How to shorten the flash memory update time?***

4.3.3 Executing Program

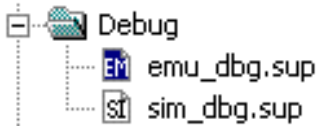
■ **Excuting program**

《 Setting procedure 》

(1) Turn on the user power to the MB2146-401.

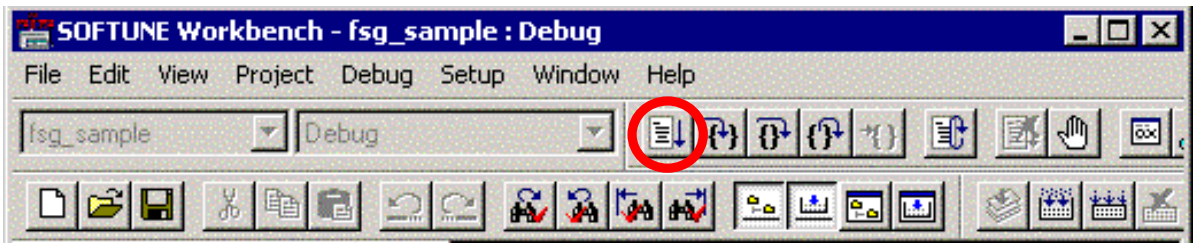
(2) The MB2146-09 POWER LED turns orange.

(3) Double-click **emu_dbg.sup** in the Debug folder in the Project window to start the Emulator Debugger.



(4) Reset MCU before executing the program.

(5) When the user makes the program run continuously, each MB2146-401 LED comes on in sequence.



(6) Check that each LED is on and then terminate the program.

4.3.4 Setting Breakpoint

■ **Setting Breakpoint**

During the Debugger executes a program, the execution of the program can be stopped when the program counter (PC) passes a certain address or accesses data at a certain address.

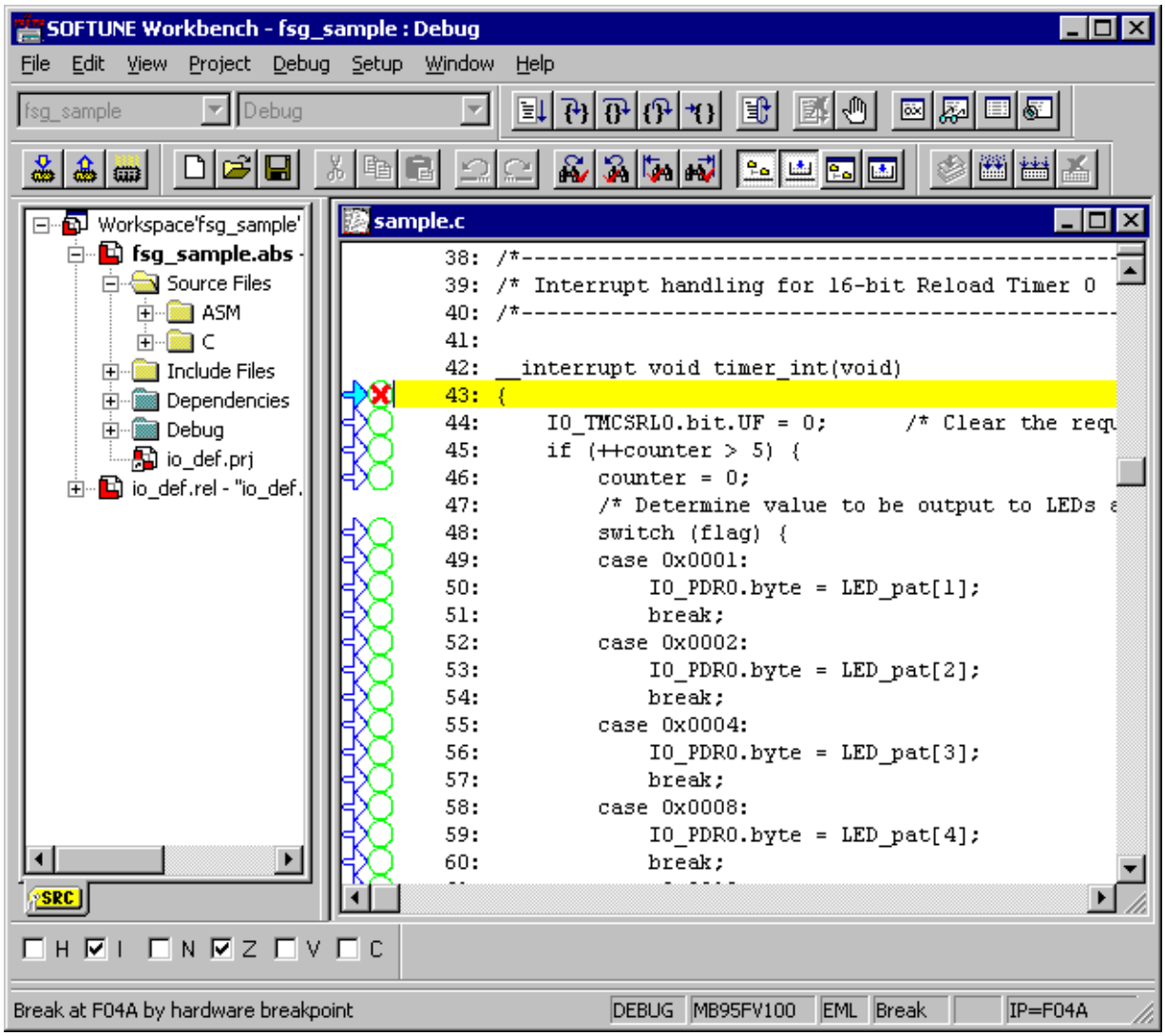
The position where the execution of the program is stopped is called a breakpoint. The user can set the code breakpoints, data monitoring breakpoints, and sequence breakpoints.

This sample sets a breakpoint at the beginning of the timer interrupt function in **`sample.c`**.

Click the left circle of the source window to set the breakpoint “X”.

The yellow line indicates the current position of the instruction pointer (IP).

Note: The IP value is the same as the PC value.



One Point !

= Types of breakpoints and how to set =

There are two types of breakpoint as follows:

Select [Breakpoint] from the [Debug] menu to open the [Break] dialog.

This dialog can also set breakpoints. It also provides a list of set breakpoints.

- Code breakpoint

  Code breakpoint is a breakpoint to stop the program when the PC attempts to execute a set address. For both Simulator and Emulator, the PC stop position is a breakpoint before execution. Up to 256 breakpoints can be set.

- Data breakpoint

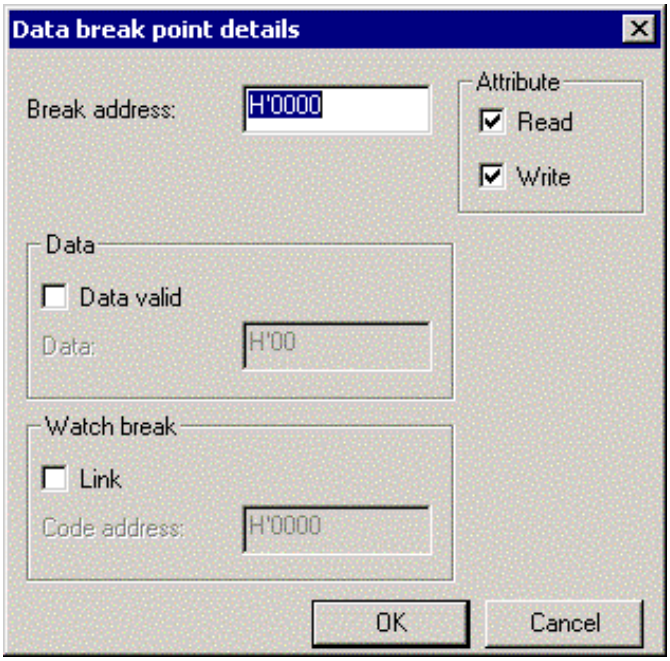
  Data breakpoint is a breakpoint to stop program when the PC accesses data at a set address.

  Read, write, and read/write can be set as access conditions. Up to two breakpoints including a data monitoring breakpoint can be set.

- Data monitoring breakpoint

  This breakpoint stops the program when the PC executes the program at a set address or accesses data at a certain address.

  Read, write, and read/write can be set as the access conditions. These conditions are enabled only when the user checks “Link” checkbox at Watch break in the “Data break point details” dialog.



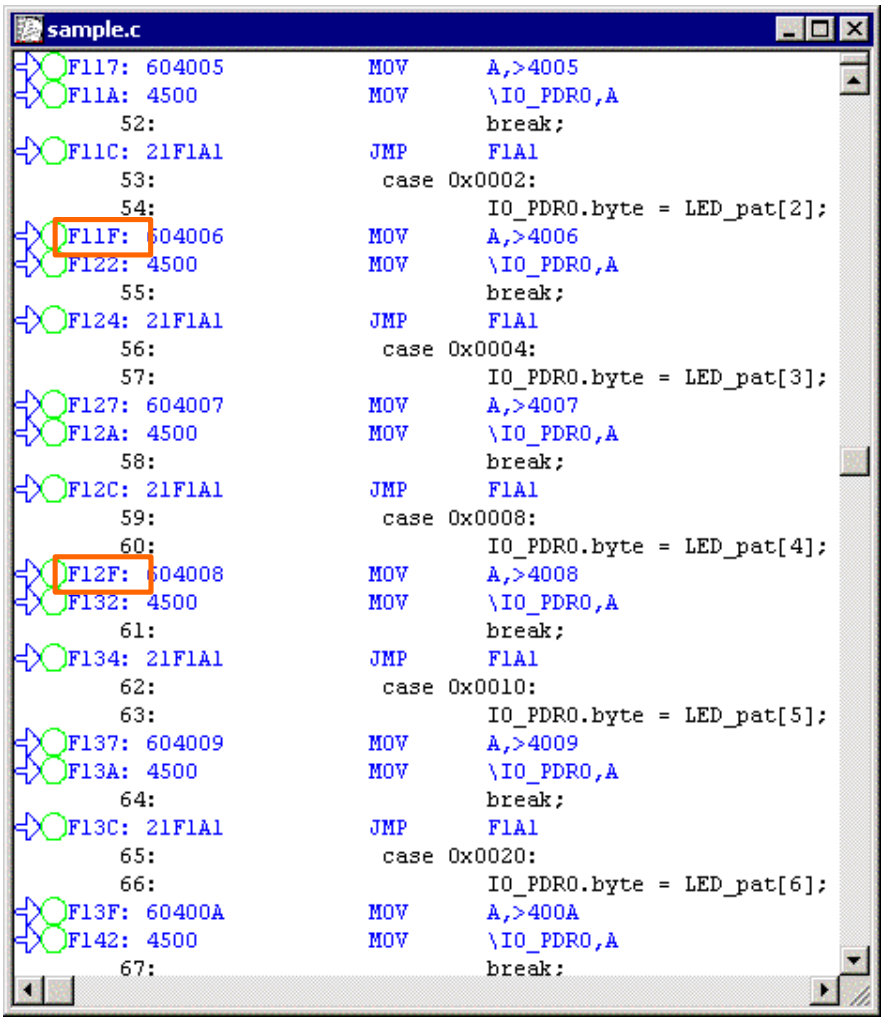
- Sequence breakpoint

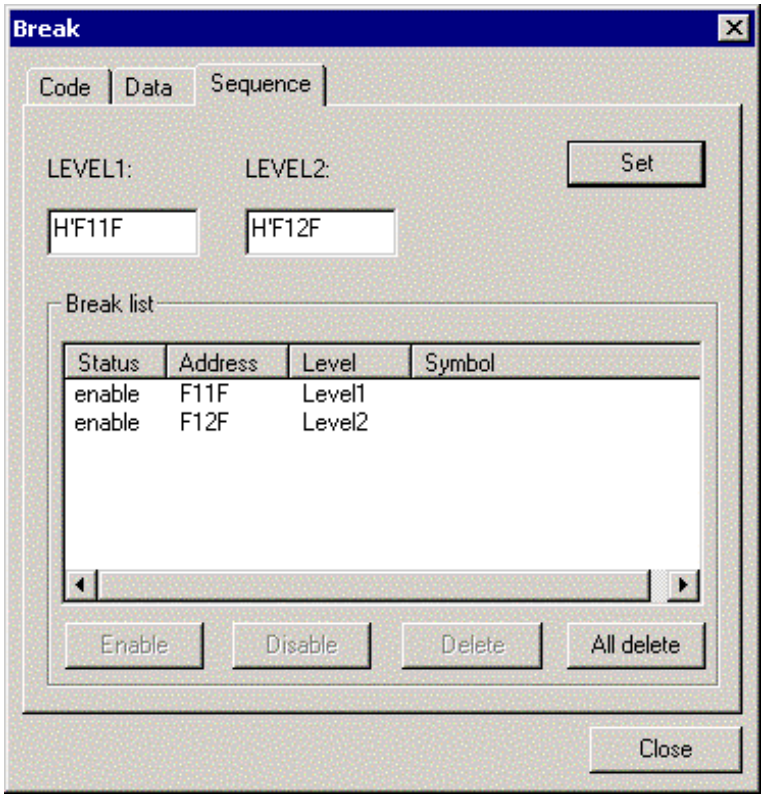
  This breakpoint stops the program being executed when the program is executed at two specified addresses from “Level 1” to “Level 2.”

《 Setting procedure 》

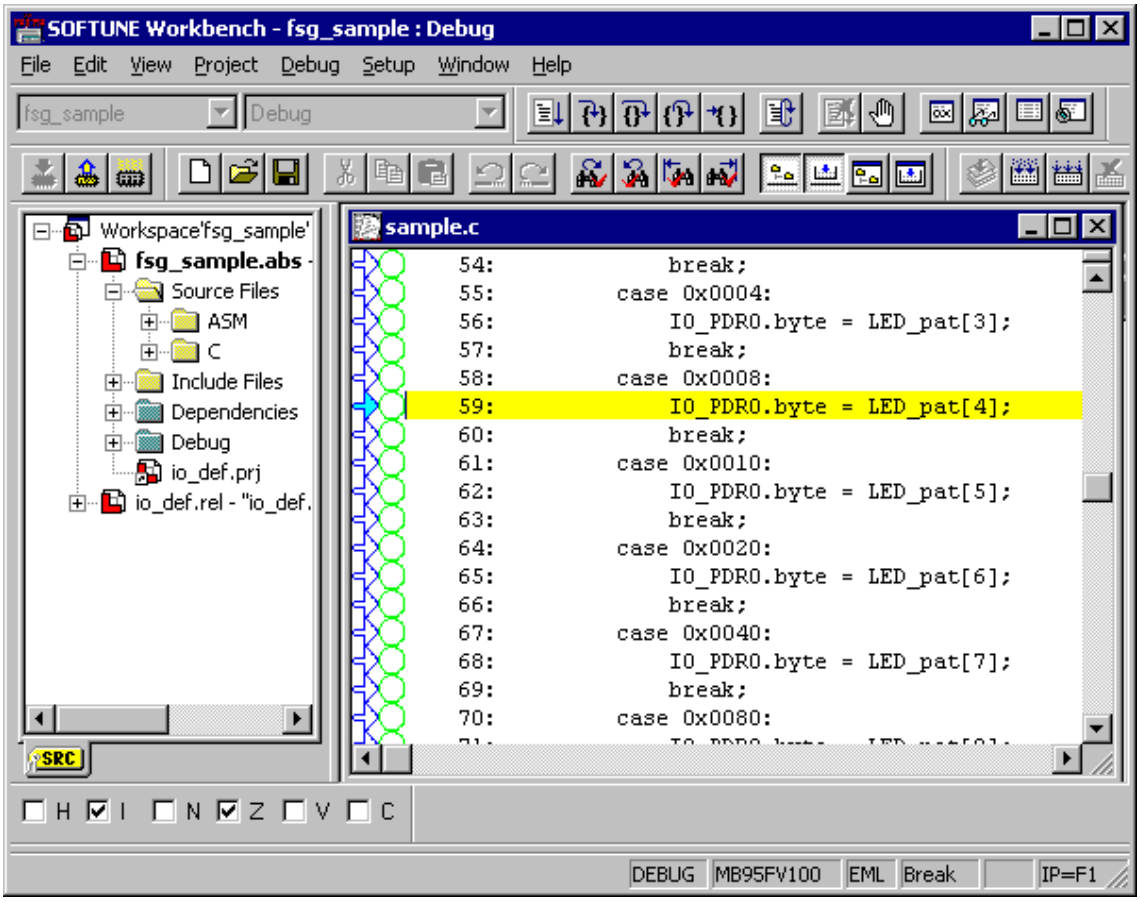
(1) Use the [Sequence] tab for [Breakpoint] in the [Debug] menu or select [Sequence] from the [Debug] menu to enter an address of the level to be set.

Select [Mix display] from the shortcut menu in the Source window, check the address, and set the level at which transition conditions are set.





(2) The program will stop when the PC passes through the sequence (from “Level 1” to “Level 2”).



4.3.5 Trace

■ **Trace**

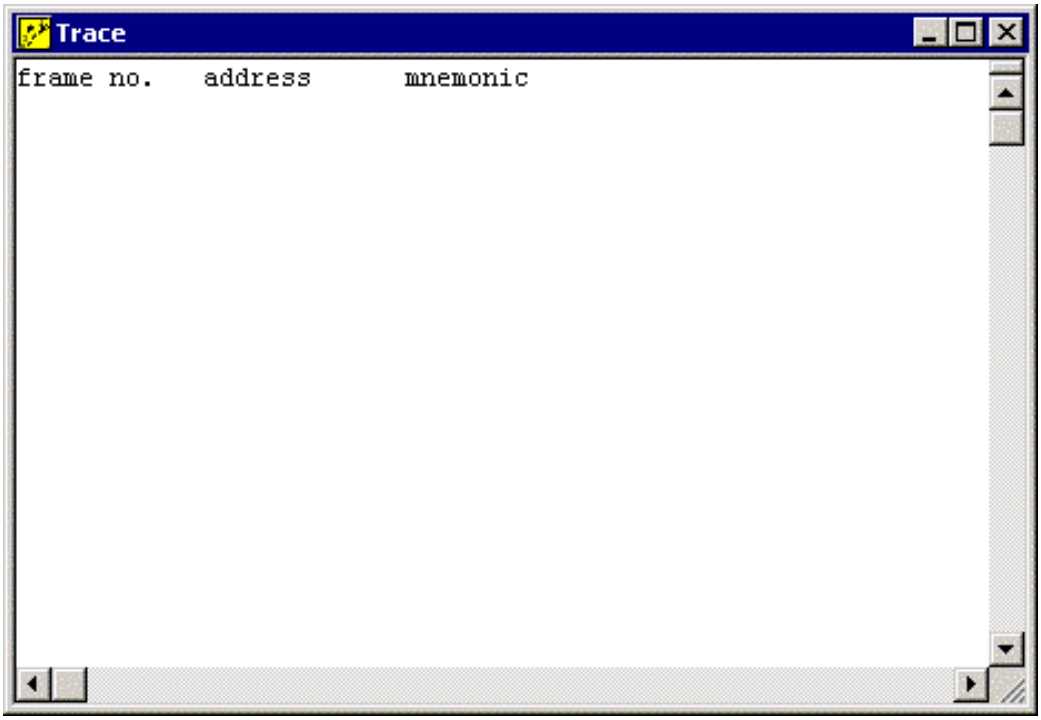
When the trace function is enabled, data is always sampled and stored in the trace buffer during execution.

Trace information to be actually displayed is that on the 16 branches immediately preceding suspension of execution.

The trace buffer is structually a ring-shaped, so data is overwritten automatically from the beginning of the sequence buffer when the trace buffer is full.

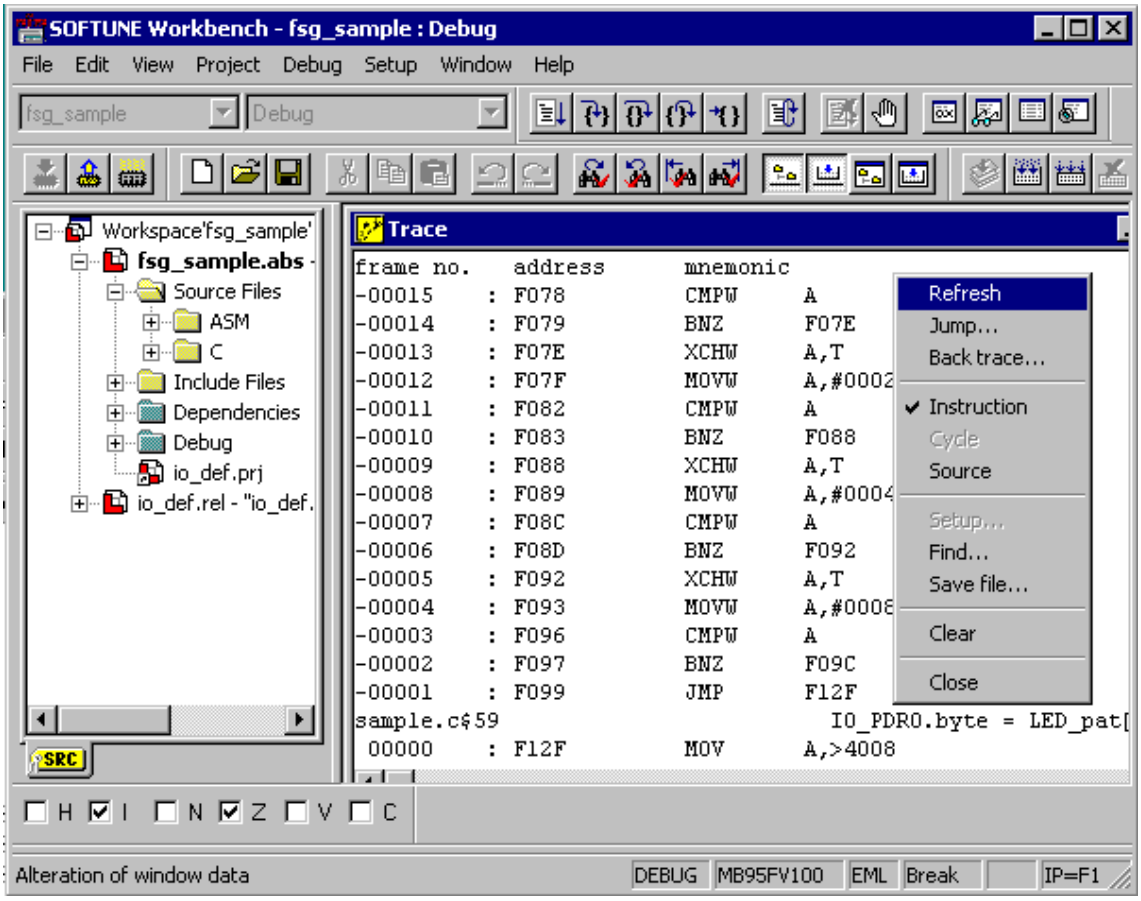
《 Setting procedure 》

(1) Select [Trace] from the [View] menu to open the Trace window.



(2) The program is executed and stopped.

Right-click the Trace window to select [Refresh] from the shortcut menu. The trace results appears.



One Point !

= Display format for trace data =

There are two display formats for trace data:

- Display in order of instruction execution (Instruction)
- Display in source lines (Source)

Use the shortcut menu to switch between the display formats for trace data.

No trace stamp is displayed.

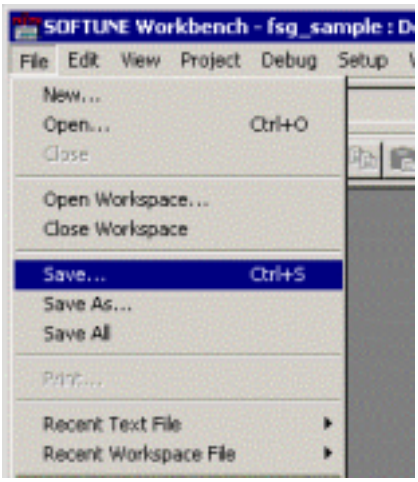
One Point !

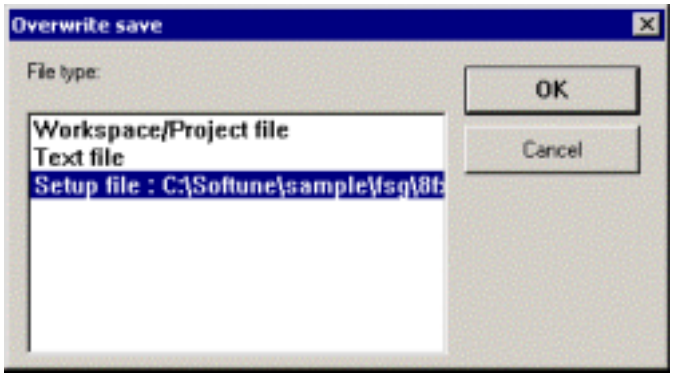
= Saving setup file for the Emulator =

A setup file is not saved automatically until debugging is terminated. When updated, save the setup file.

Select the [Save...] command from the [File] menu to save the setup file for the Emulator in the Save dialog.

The setup file can be saved only in the debug mode.





4.4 Operating Object Format Converter

■ Operating Object Format Converter

This section explains the procedure for making a new project configuration for release separately from the project configur for debugging and how to operate the object type Converter.

The Object Format Converter handles the following file formats:

Absolute format load module for Linker output	(~**.abs**)
Motorola S format	(~**.mhx**/~**.ahx**)
Intel HEX format	(~**.ihx**/~**.ehx**/~**.hex**)
Binary data file	(~**.bin**)

4.4.1 Adding Project Configuration

■ Adding Project Configuration

A project configuration is a function to manage files for Make/Build or the conditions for different option settings with one project.

This function allows making and managing FLASH microcontroller project to which FLASH self-rewritable program files are Member-added, and mask release projects that remove FLASH self-rewritable program files from Make/Build operation. The project configuration can be switched to make objects and debug for any purpose. The default project configuration name "**Debug**" is made when making a new project. When making an additional project configuration, the project configuration with the same settings as those of the existing project configuration is copied. Removing unnecessary program files in this project configuration from Make/Build operation or resetting for Make/Build operation, a new project configuration can be easily made.

Instructions are given for the procedure for making the following two project configurations:

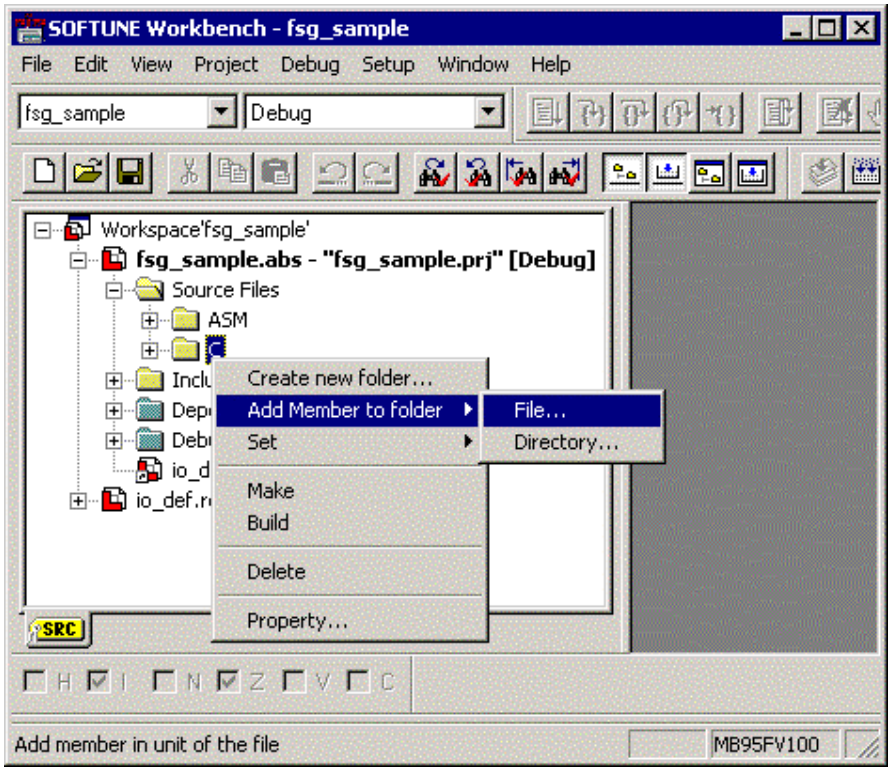
- FLASH microcontroller project configuration (flash)
- Mask release project configuration (mask)

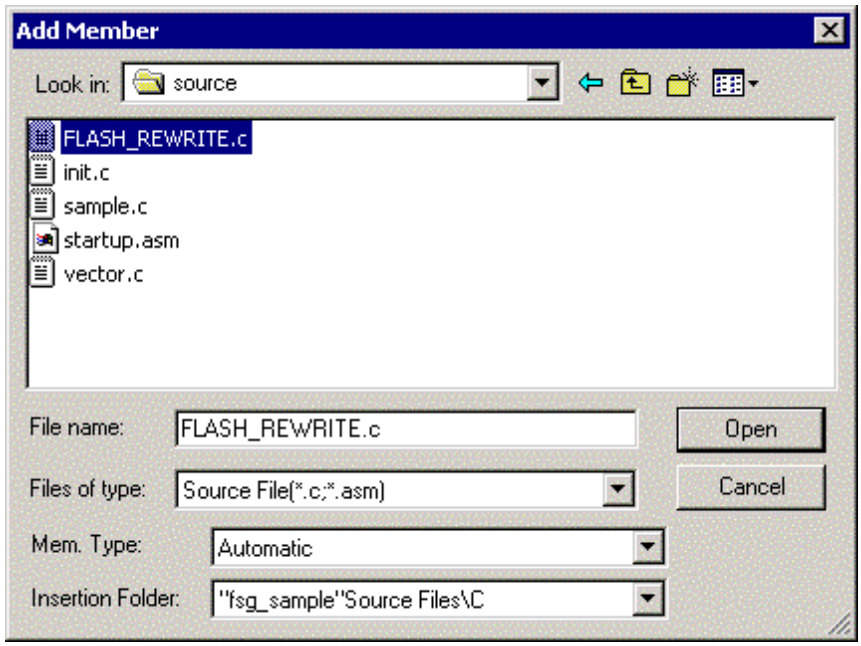
The following file is used just to explain the procedure. This file is for the instructions, and does not contain any program.

- **FLASH_REWRITE.c** (FLASH rewritable program)

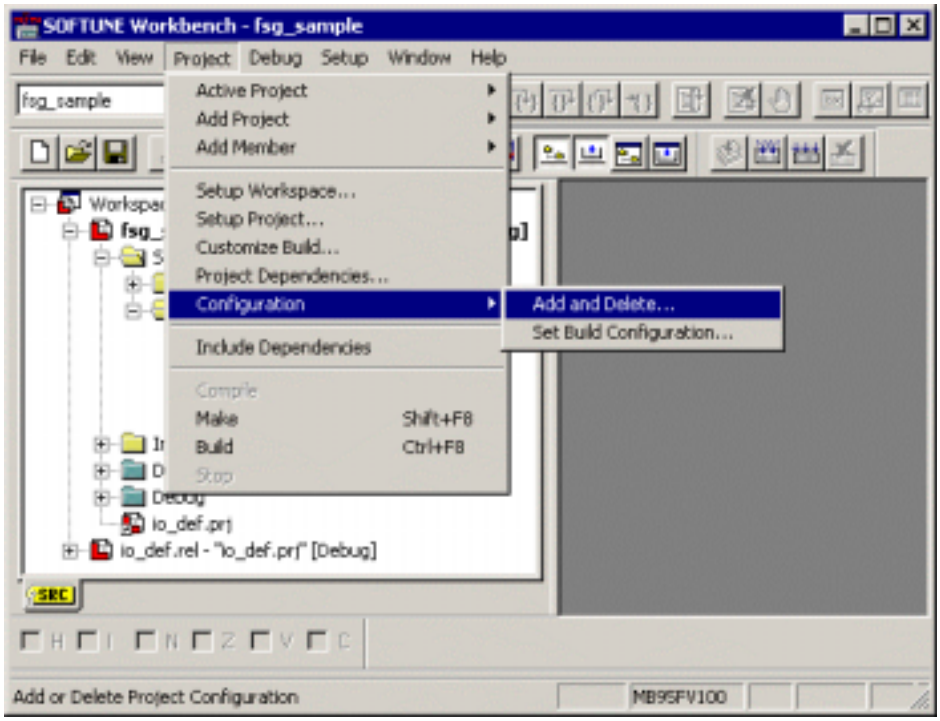
《 Setting procedure 》

(1) Select the C folder in the Project window, select [Add Member to folder]-[File] from the shortcut menu to add **FLASH_REWRITE.c** to the Member.



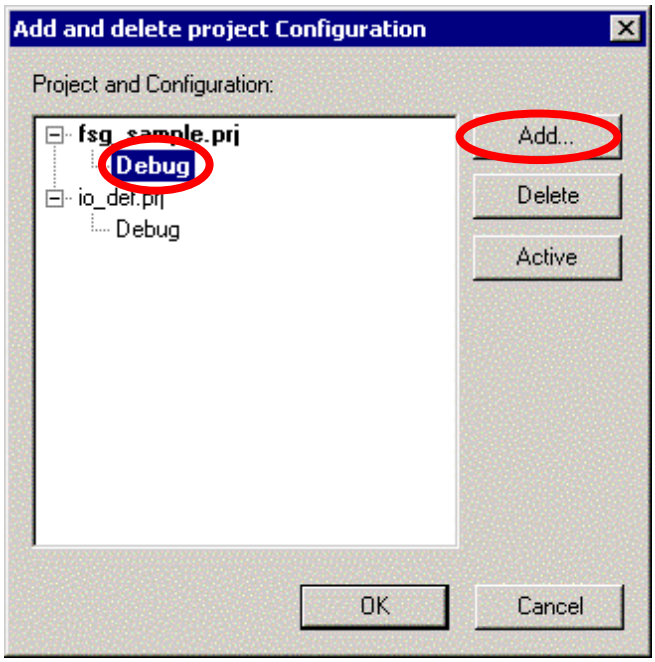


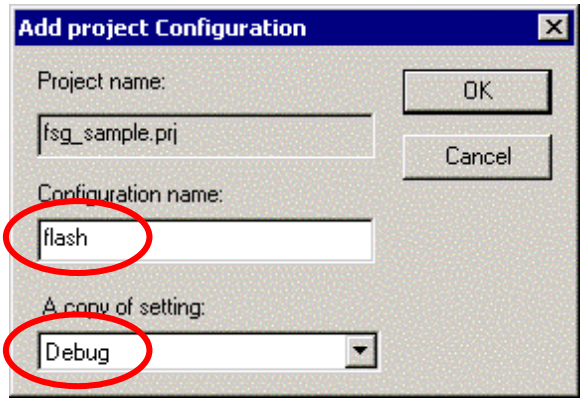
(2) Select [Configuration]-[Add and Delete...] from the [Project] menu.



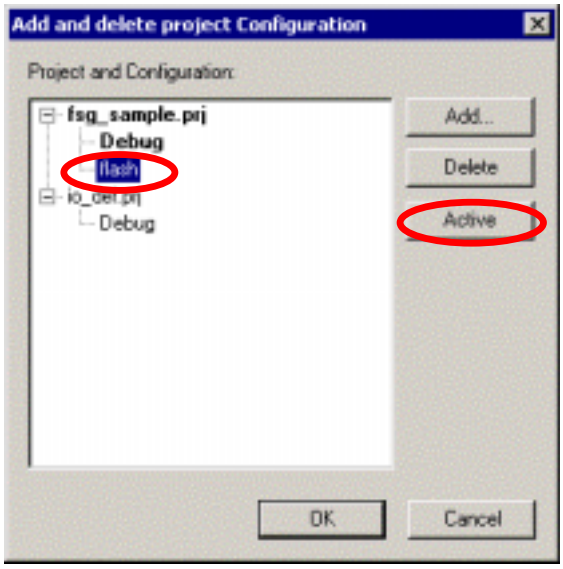
(3) Click **Debug** in **fsg_sample.prj** in the Add and delete project configuration dialog and click [Add...].

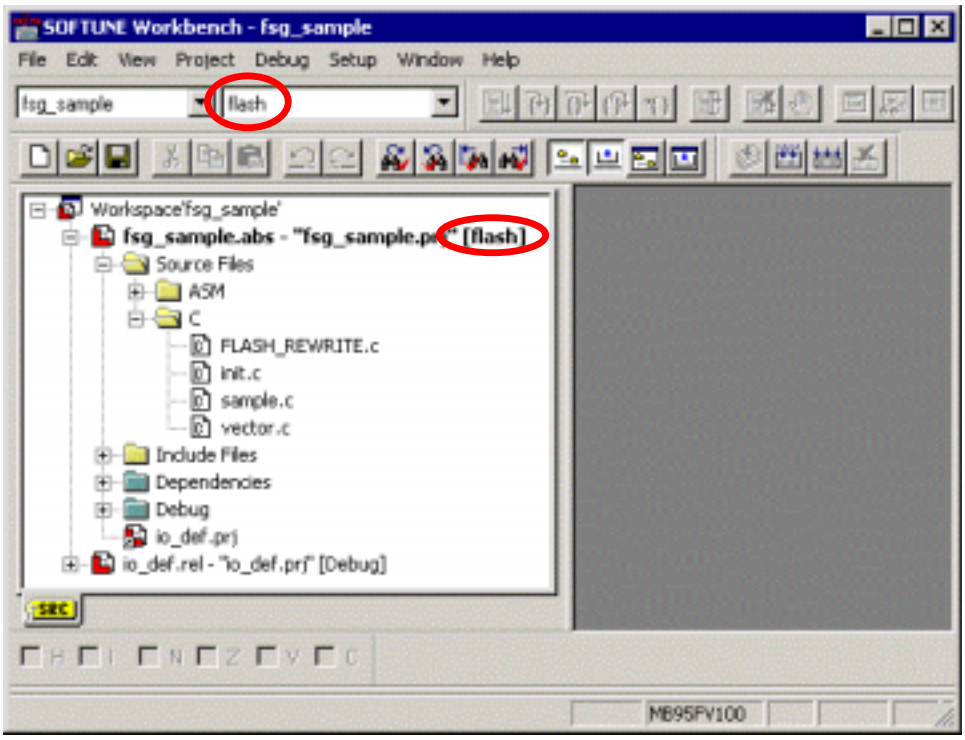
Set **flash** as a new project name in the Add and delete project configuration dialog.



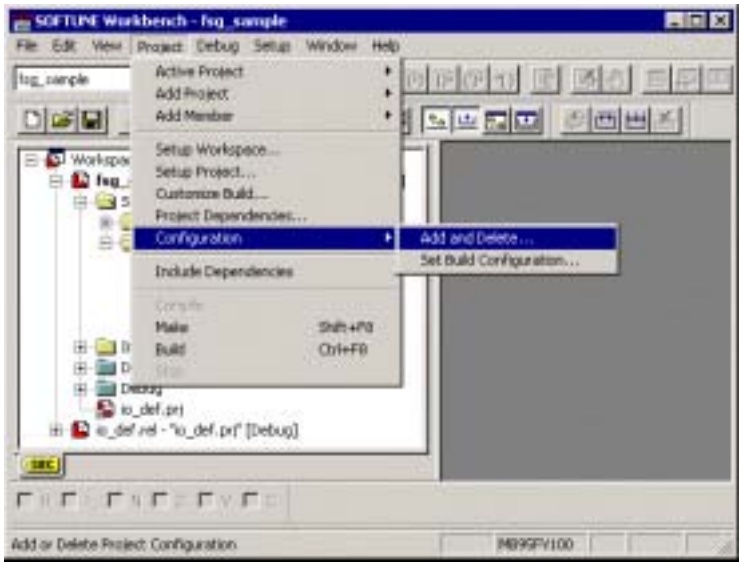


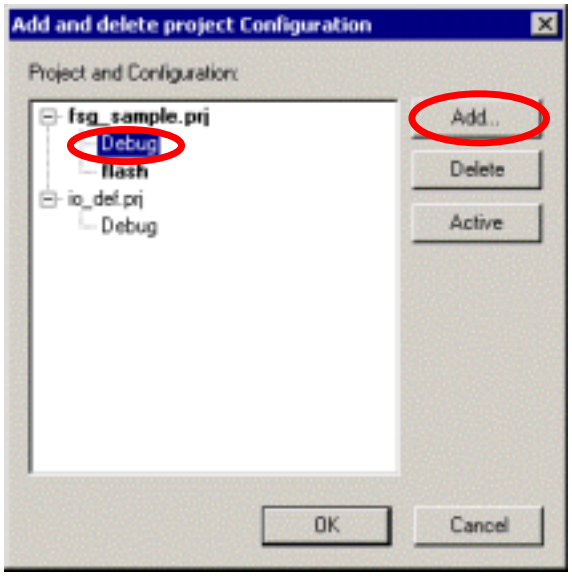
(4) Click the added project configuration **flash** and click [Active]. Check that the active project configuration is **flash** in the ***SOFTUNE*** Project window.

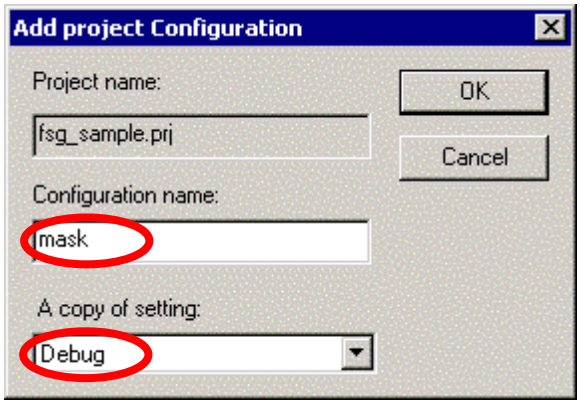


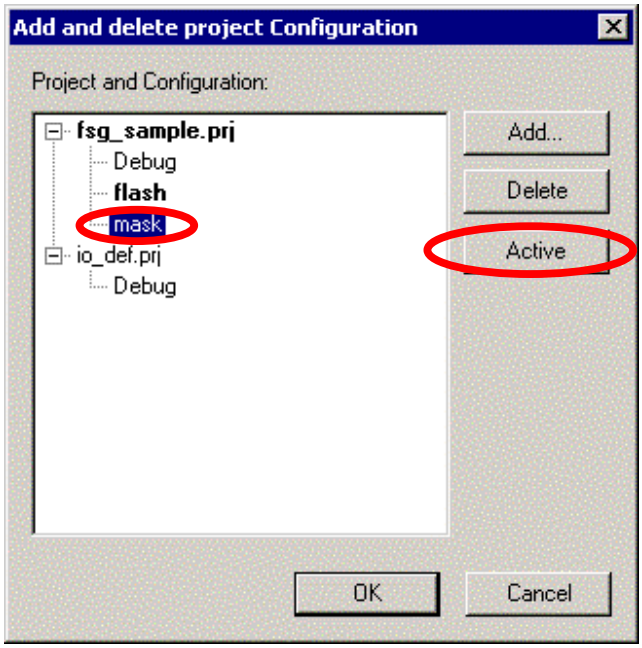

SOFTUNE Workbench - fsg_sample
File Edit View Project Debug Setup Window Help
fsg_sample
flash
Workspace'fsg_sample'
fsg_sample.abs - "fsg_sample.prj" [flash]
Source Files
ASM
C
FLASH_REWRITE.c
init.c
sample.c
vector.c
Include Files
Dependencies
Debug
io_def.prj
io_def.rel - "io_def.prj" [Debug]
SRC
MB95PV100

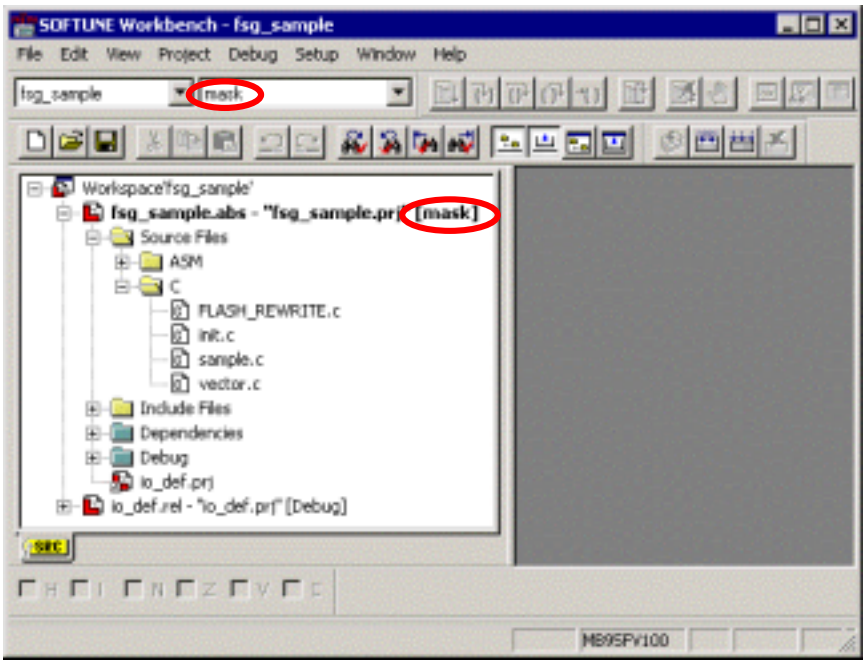
(5) Follow steps (2) to (4) to set **mask** as a new project configuration name.





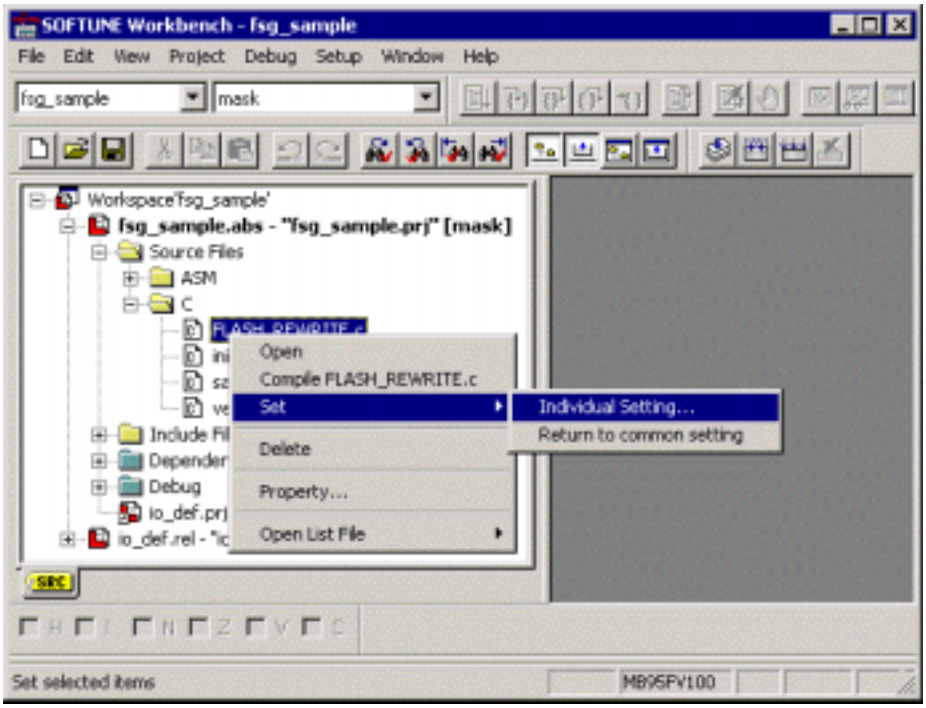

Add project Configuration
Project name:
fsg_sample.prj
OK
Cancel
Configuration name:
mask
A copy of setting:
Debug


Add and delete project Configuration
Project and Configuration:
fsg_sample.prj
Debug
flash
mask
io_def.prj
Debug
Add...
Delete
Active
OK
Cancel

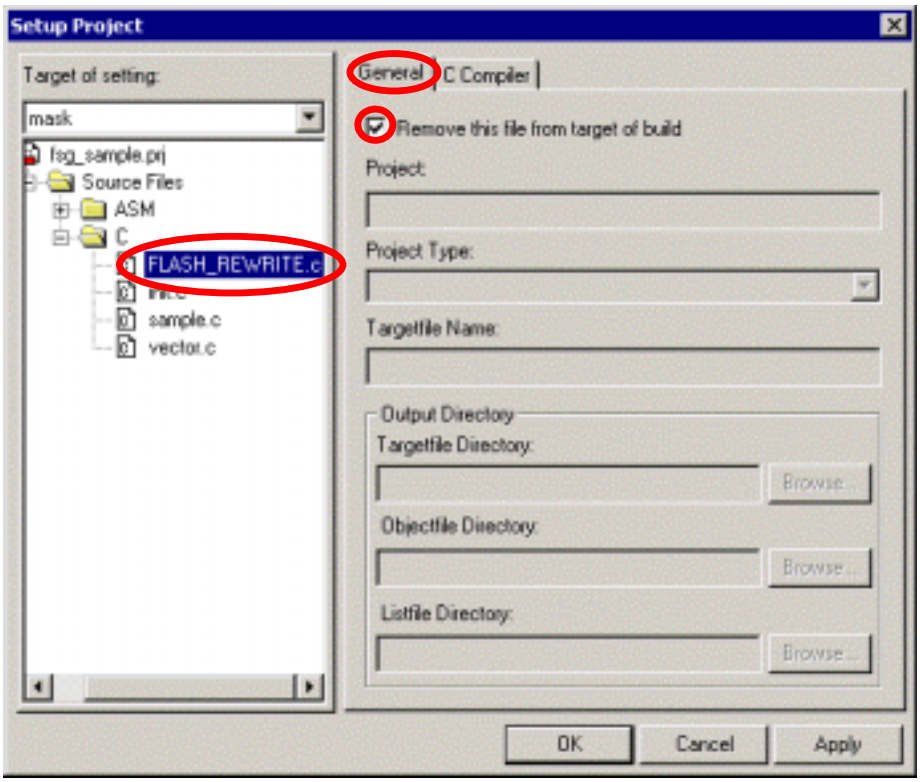

SOFTUNE Workbench - fsg_sample
File Edit View Project Debug Setup Window Help
fsg_sample
mask
Workspace'fsg_sample'
fsg_sample.abs - "fsg_sample.prj" [mask]
Source Files
ASM
C
FLASH_REWRITE.c
init.c
sample.c
vector.c
Include Files
Dependencies
Debug
io_def.prj
io_def.rel - "io_def.prj" [Debug]
MB95FV100

(6) The added project configuration **mask** does not require **FLASH_REWRITE.c**. Remove it from Make/Build operation.

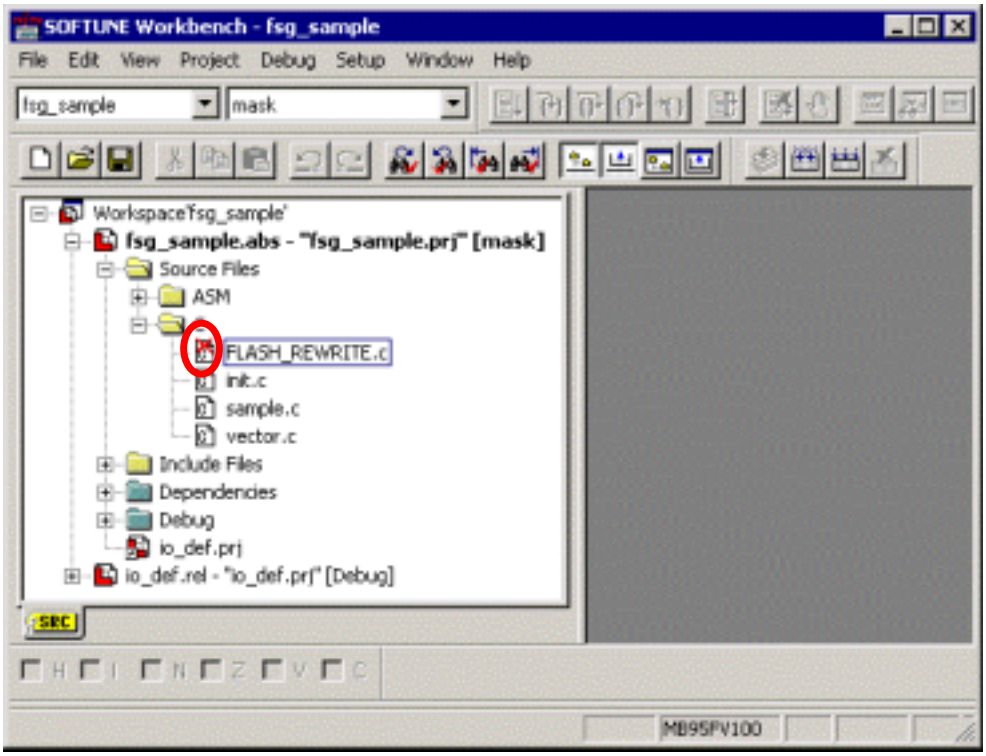
Click **FLASH_REWRITE.c** in the C folder in the Project window and select [Set]-[Individual setting...] from the shortcut menu.



(7) Click **FLASH_REWRITE.c** in the Setup project dialog and check the “Remove this file from target of build” checkbox.



(8) In the ***SOFTUNE*** Project window, check that **`FLASH_REWRITE.c`** is marked with “×” (that means the file is removed from Make/Build operation).



4.4.2 Operating Converter

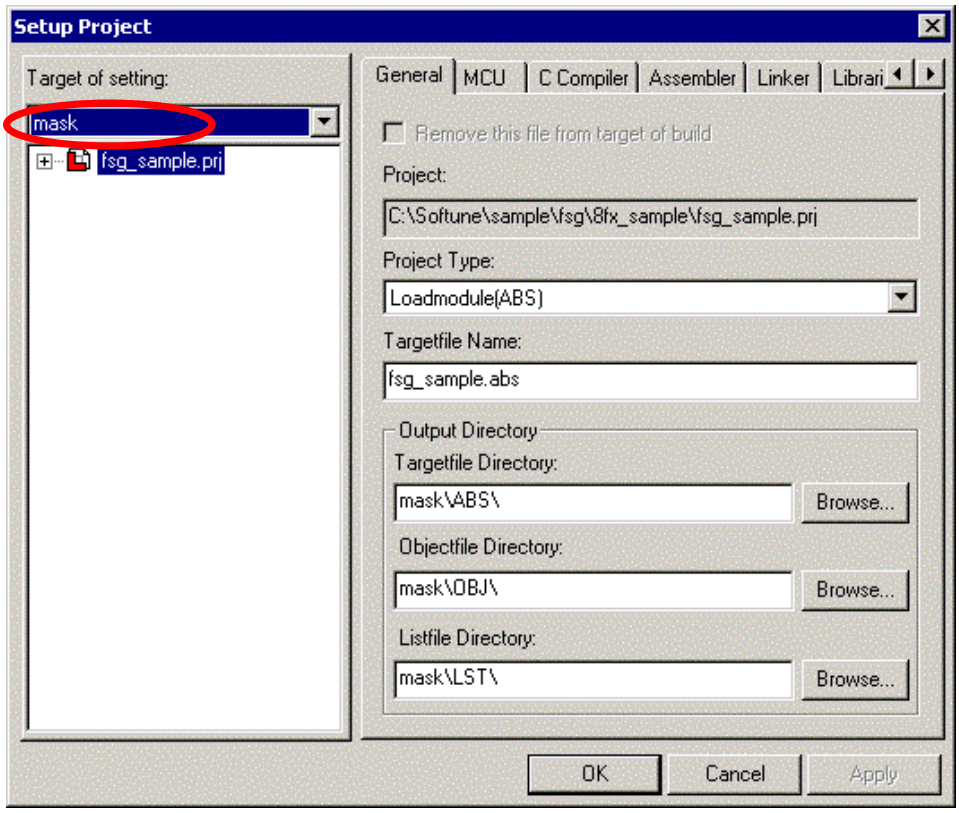
■ **Operating Converter**

This section explains the procedure for operating the object type Converter by the newly added project configuration mask.

This sample converts the created target file (**`fsg_smple.abs`**) to the Motorola S format (**`fsg_sample.mhx`**).

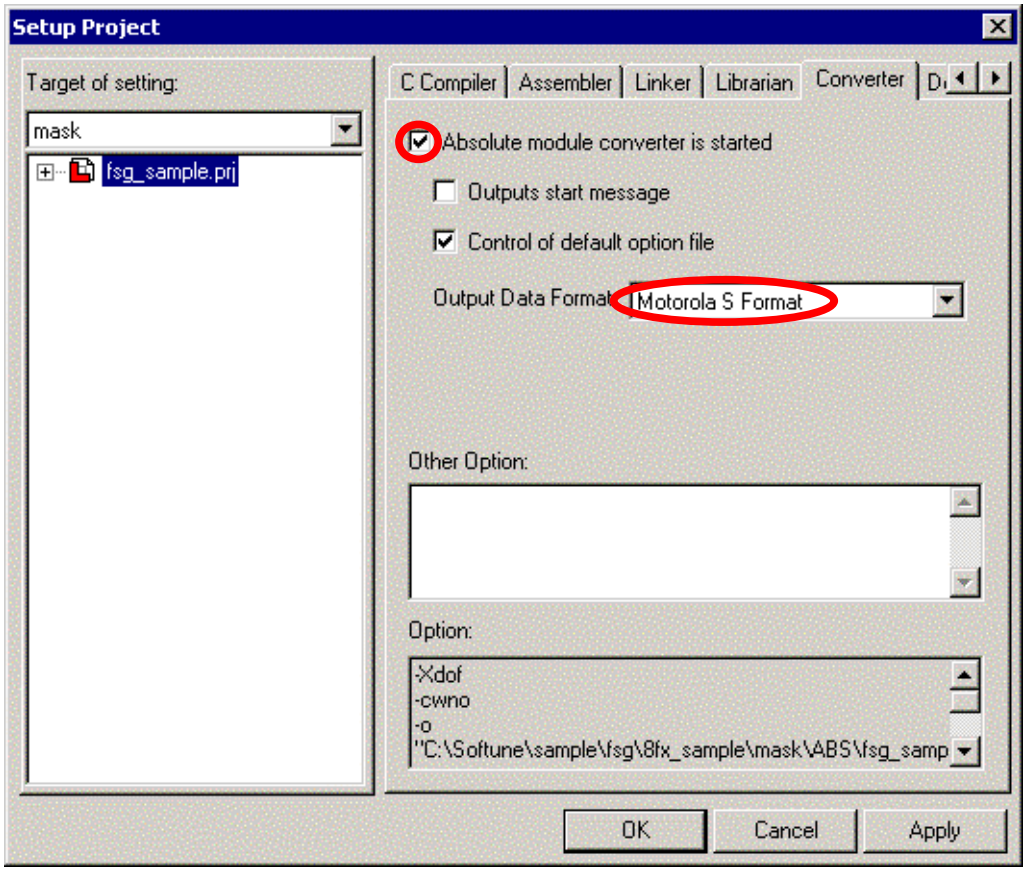
《 Setting procedure 》

(1) Select [Setup project] from the [Project] menu and check that Target of setting is “mask.”

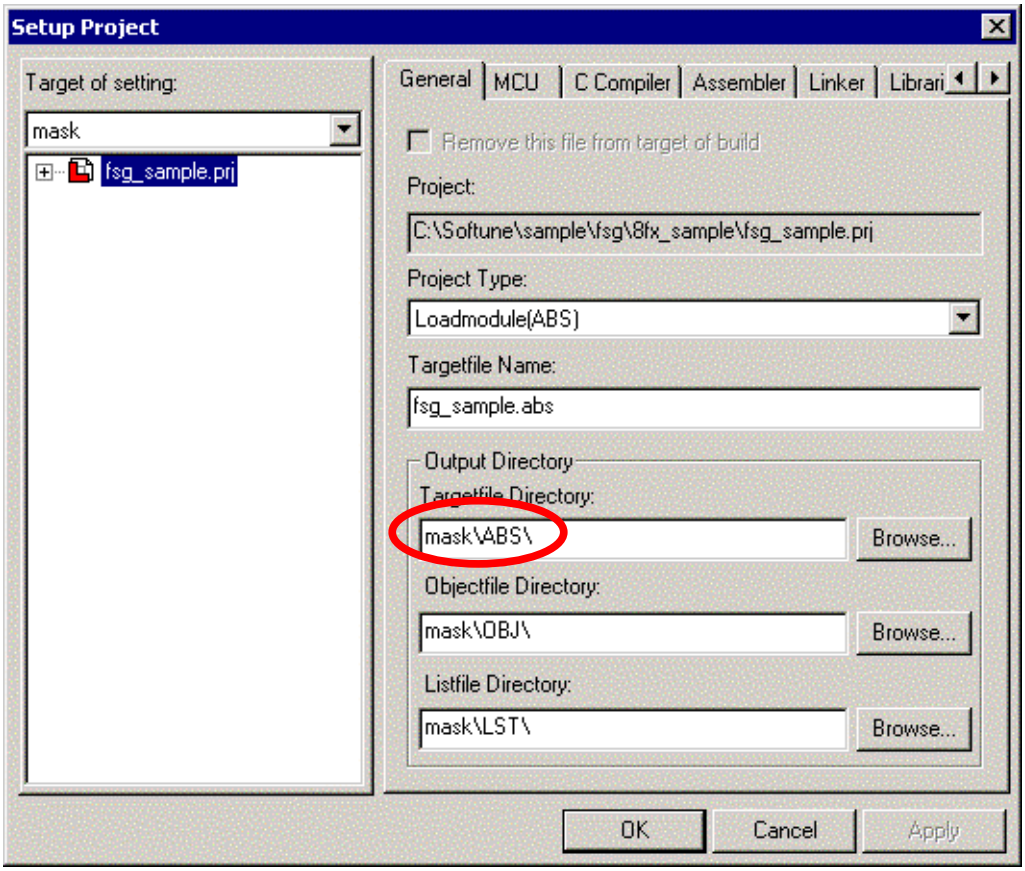


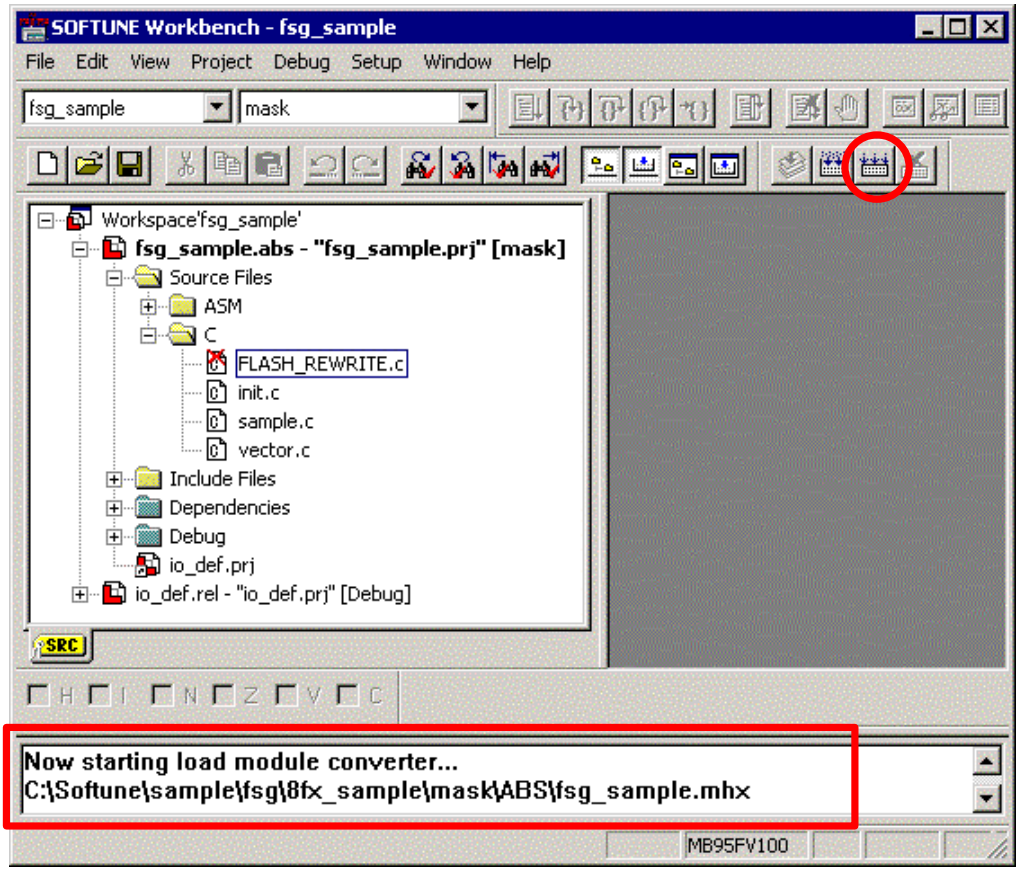
(2) Specify the Converter format to be converted.

In the Converter tab, check the [Absolute module converter is started] checkbox and select the Motorola S format as the Output Data Format.



(3) When [Build] is executed, the Converter is started after the target file is created, generating the conversion file (**`fsg_sample.mhx`**) in the target directory.




SOFTUNE Workbench - fsg_sample
File Edit View Project Debug Setup Window Help
fsg_sample
mask
Workspace'fsg_sample'
fsg_sample.abs - "fsg_sample.prj" [mask]
Source Files
ASM
C
FLASH_REWRITE.c
init.c
sample.c
vector.c
Include Files
Dependencies
Debug
io_def.prj
io_def.rel - "io_def.prj" [Debug]
SRC
Now starting load module converter...
C:\Softune\sample\fsg\8fx_sample\mask\ABS\fsg_sample.mhx
MB95FV100

5. USEFUL FUNCTIONS

5.1 Setting External Tool

■ **Merit in Setting External Tool**

Registering various tool commands allows the users to start commands from the ***SOFTUNE*** Workbench desktop. The following setting example registers the object type Converter (S format adjuster), and its usage example converts the Motorola S format (**`fsg_sample.mhx`**) made in Section ***4.4.2***.

= What is the S format Adjuster? =

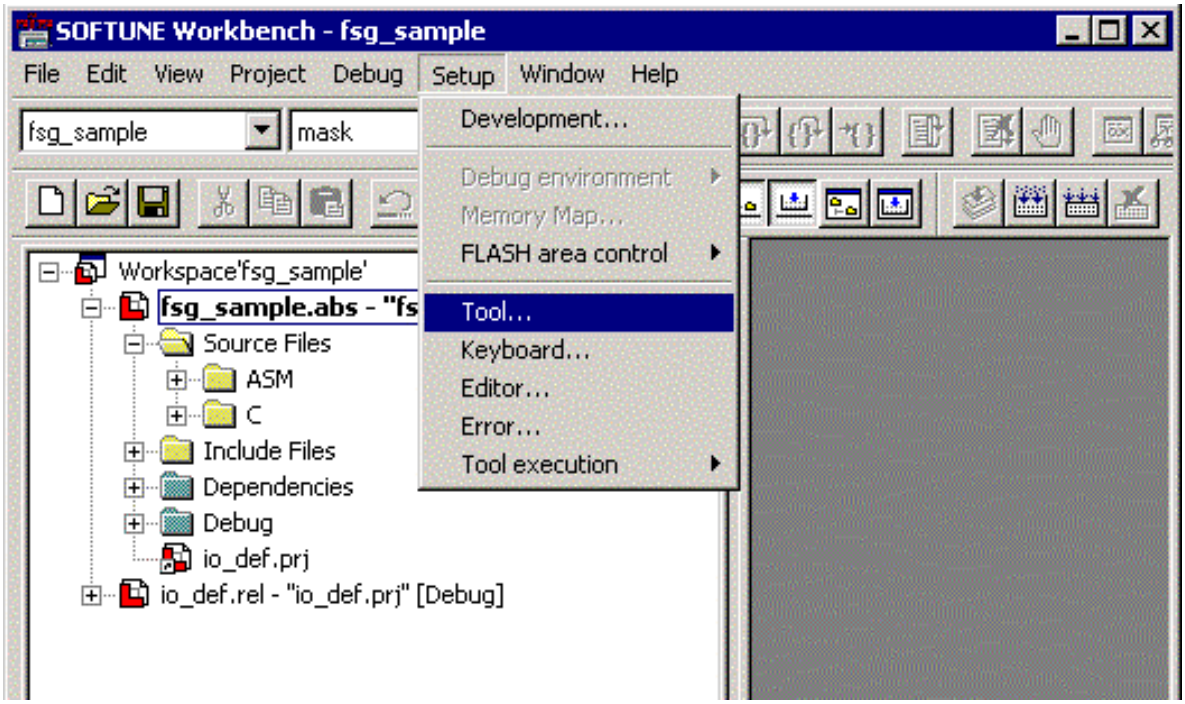
The S format Adjuster adjusts data made in Motorola S format in ascending address order and aligns the data count included in one record on a specified value.
Using the S format Adjuster, the formatted file can be made with its Motorola S format kept.

For details of the Adjuster, refer to Help (***Part 4 Object Format Converters of SOFTUNE Linkage Kit***)

■ Setting and Using External Tool

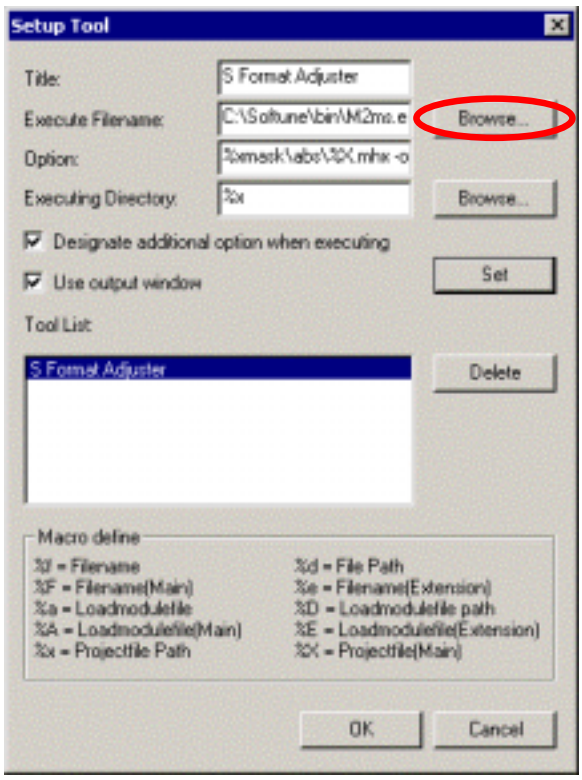
《 Setting procedure 》

(1) Select [Tool...] from the [Setup] menu.



(2) Set the following items in the [Setup Tool] dialog.
Check the “Designate additional option when executing” and “Use output window” checkboxes.
Click the [Set] button to set the S format Adjuster in Tool List and click the [OK] button.
Macro define can be used for the option.
In this example, a file is made in the ABS folder in the Project directory where the load module is made.

- Title → S format Adjuster
- Execute Filename → C:\Softune\bin\M2ms.exe
- Option → %xmask\abs\%X.mhx -o %xmask\abs\%X.ahx
- Executing Directory → %x

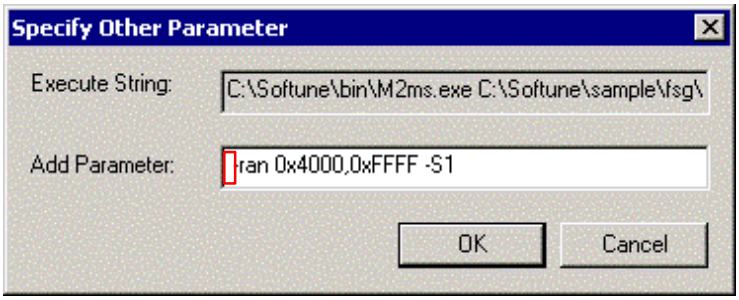


This completes the tool setting.

This sample continues to make the formatted Motorola S format file (**`fsg_sample.ahx`**) using the S format Adjuster.

(3) Select [Too execution]-[1 S Format Adjuster] from the [Setup] menu and start the registered tool.

(4) When the [Specify Other Parameters] dialog is opened, specify the parameters. Be sure to put one space before each parameter.



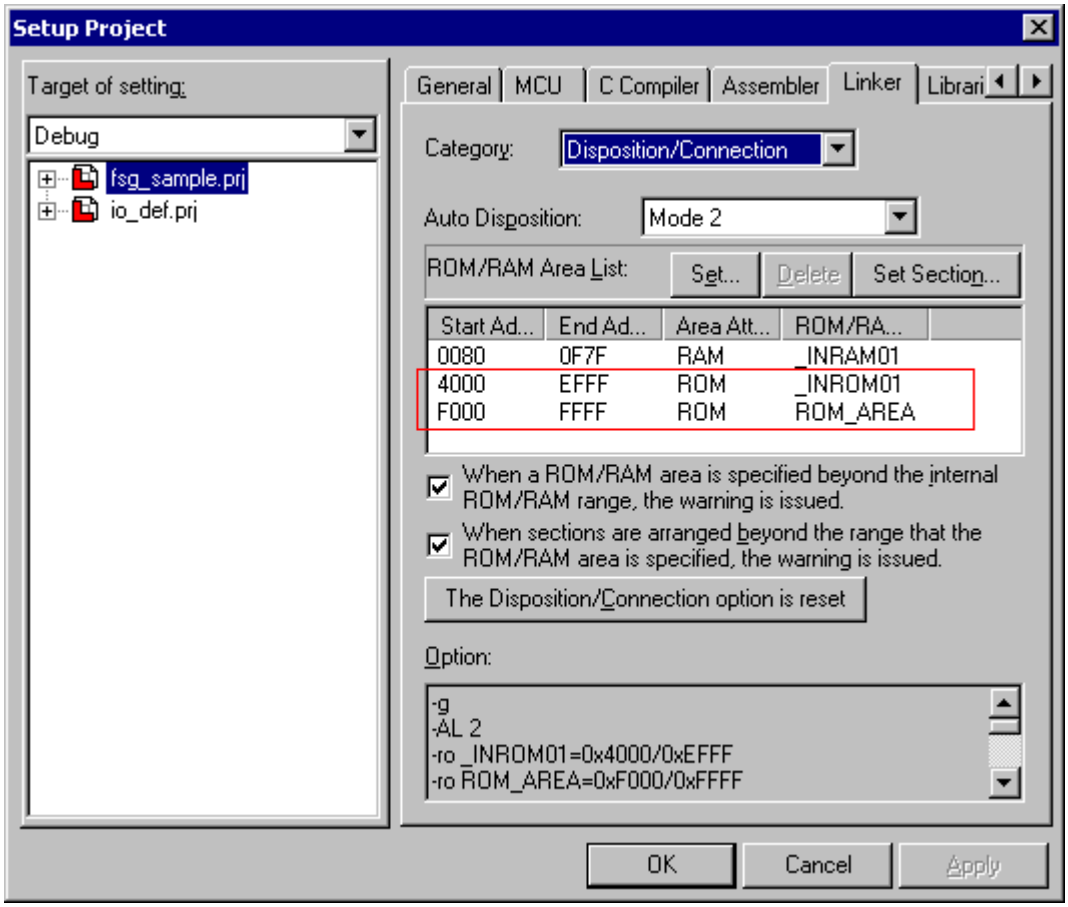
Specify the following two items as parameters:

- Output range (**-ran**)
- Output record (**-S1**/**-S2**/**-S3**)

= Output range (-ran) =

Specify the ROM area of the load module as a parameter.

To check whether the ROM area is specified, select [Setup Project] from the [Project] menu and choose “Disposition/Connection” at Category in the Linker tab.



= Output record (-S1/-S2/-S3) =

The output record depends on the ROM range.

In the MB965FV100, the ROM area ranges from 0x0000 to 0xFFFF and “-S1” is specified.

Setting	Output range
-S1	0x0000 to 0xFFFF
-S2	0x000000 to 0xFFFFFF
-S3	0x00000000 to 0xFFFFFFFF

(5) The S format Adjuster is started to make the formatted Motorola S format file.

(C:\Softune\Sample\fsg\8x_sample\mask\ABS\fsg_sample.ahx)

5.2 Setting External Editor

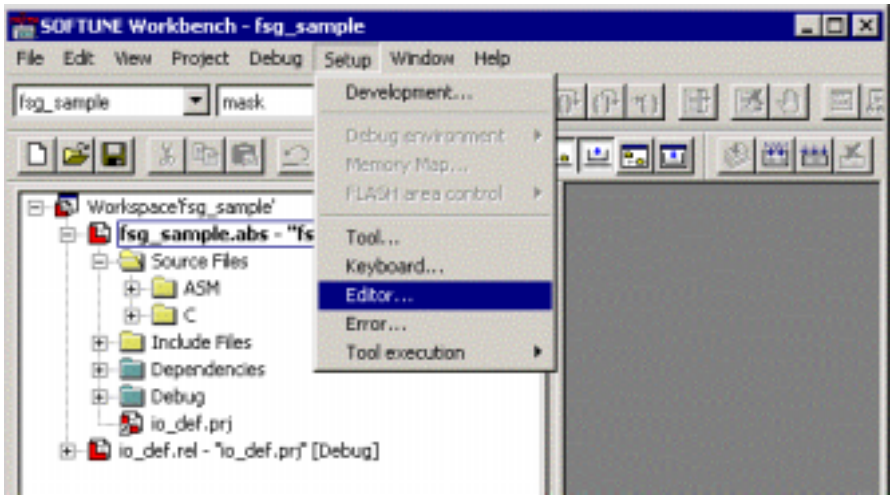
■ Merit in using External Editor

Instead of the internal Editor, the user can use favorite Editor.
It is also possible to start the source file entered in the Project and cause an error jump.
An example of setting WZ Editor V4.0 is given.

■ Setting Procedure for External Editor

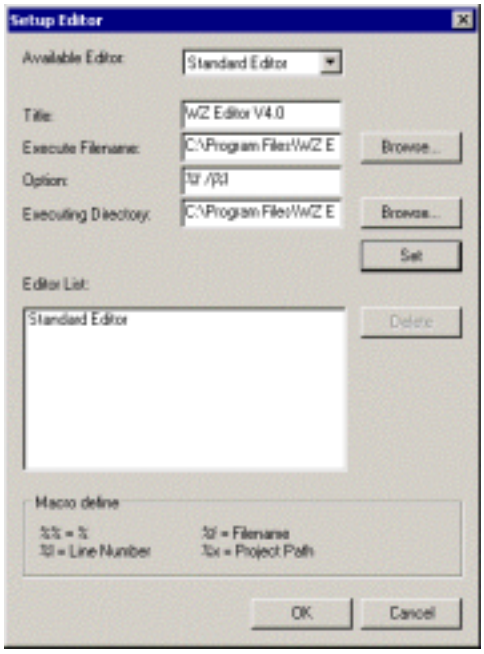
《 Setting procedure 》

(1) Select [Editor...] from the [Setup] menu.

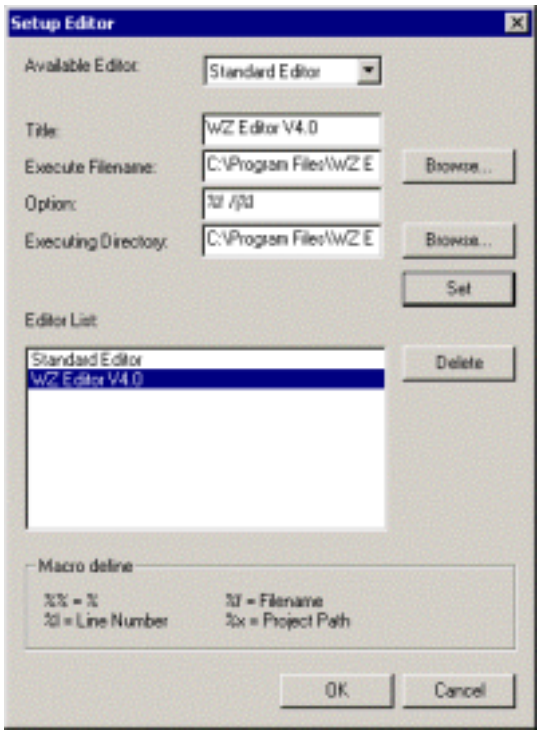


(2) Set the following items in the [Setup Editor] dialog.

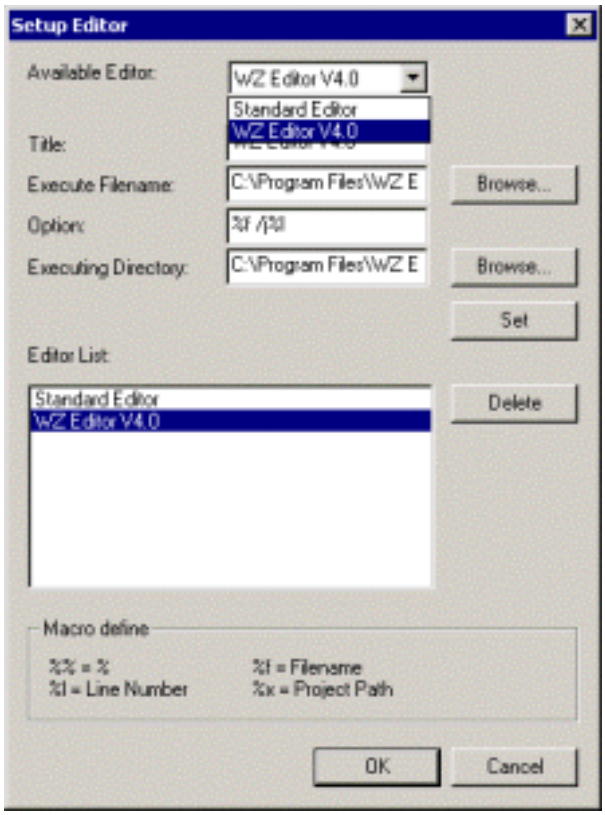
- Title → WZ Editor V4.0
- Execute Filename → C:\Program Files\WZ EDITOR\Wzeditor.exe
- Option → %f /j%l
- Executing Directory → C:\Program Files\WZ EDITOR\



(3) Click the [Set] button to enter the editor in Editor List.



(4) Select “WZ Editor V4.0” for Available Editor and click the [OK] button.



One Point !

= Caution for using external Editor =

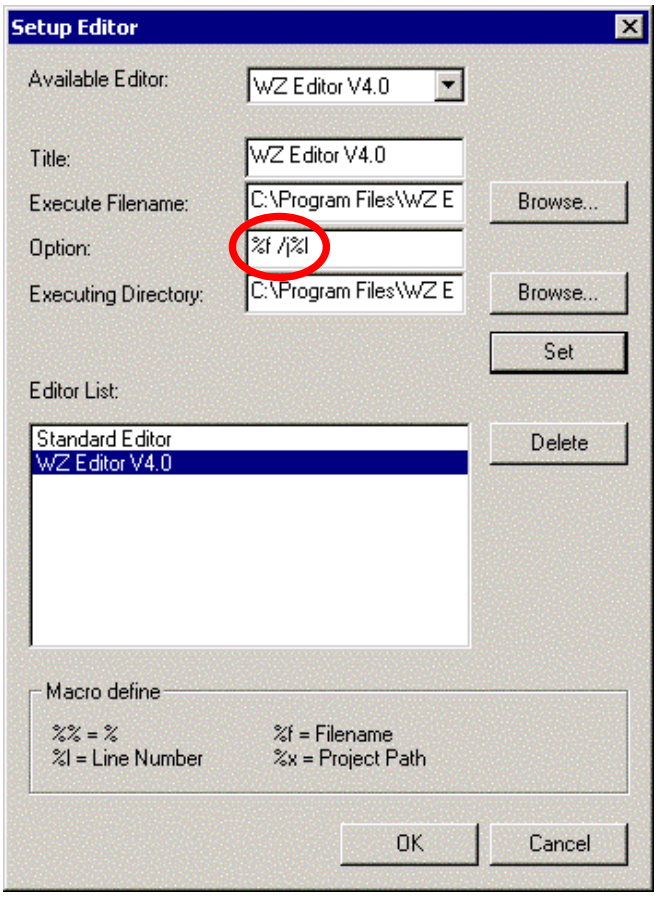
When using an external editor, a message for checking that the file is saved is not displayed at compiling or assembling even when the source file is changed.
Execute compile/assemble after saving the file.

One Point !

= Example of setting external Editor options =

- WZ Editor V4.0 → %f /j%l
- MIFES V1.0 → %f+%l
- UltraEdit32 → %f/%l/1 (English version)
- TextPal32 → %f(%l)
- Power EDITOR → %f –g%l
- Codewright32 → %f –g%l (English version of PowerEDITOR)
- Hidemaru for Win3.1/95 → /j%l:1 %f
- ViVi → /line=%1 %f

Note: Error jump by Hidemaru
To use Hidemaru as an external Editor to cause an error jump, select ‘Open same file with Hidemaru’ from the Hidemaru setup menu [Others]-[Operating Environment]-[Exclusive Control] command and set ‘Inhibit opening of same two files.’



5.3 Setting Customize Bar

■ Merit in Setting Customize Bar

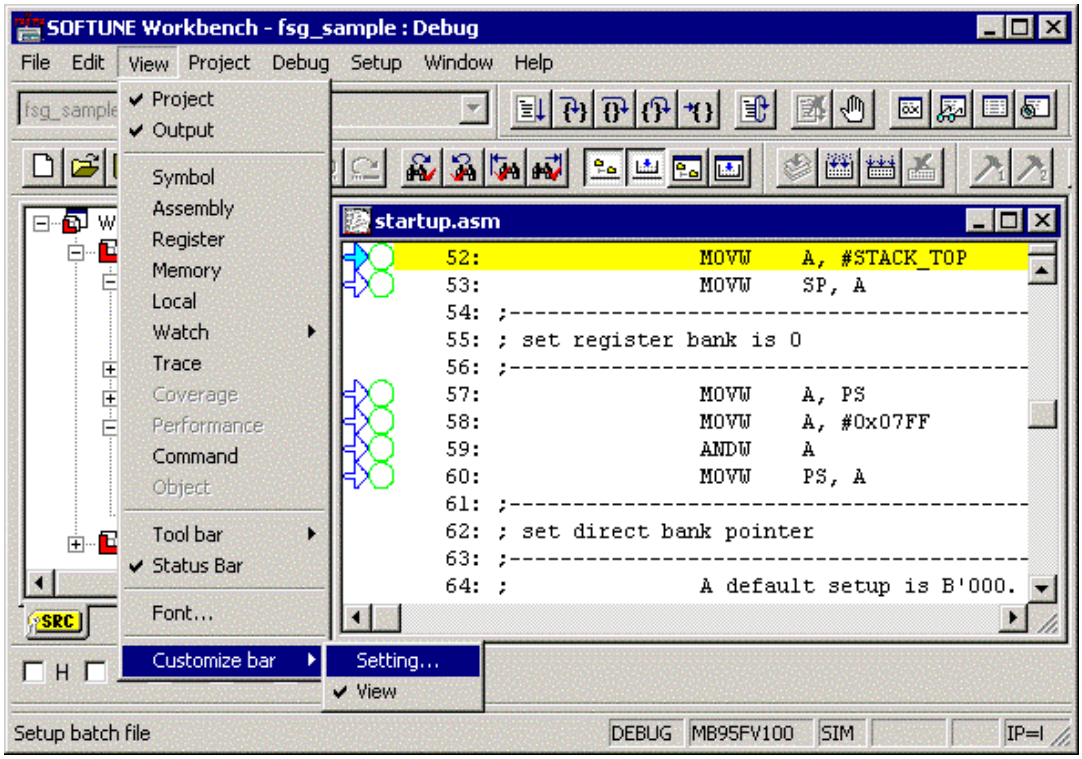
Setting the customize bar allows the user to execute any batch file at the touch of a button.
The customize bar must be installed together with F^{2}MC-8L/8FX Family SOFTUNE Professional Pack.
Here are the description of entering any batch file as an example. Up to ten customize bars can be entered.

■ Setting Procedure for Customize Bar

《 Setting procedure 》

(1) Switch to the debug mode.

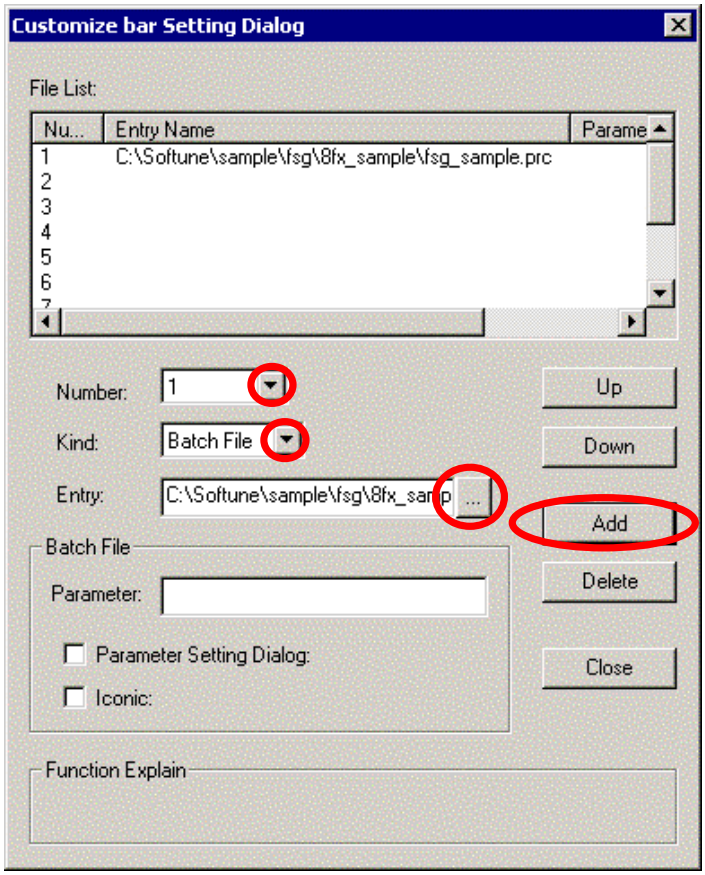
(2) Select [Customize bar]-[Setting...] from the [View] menu..



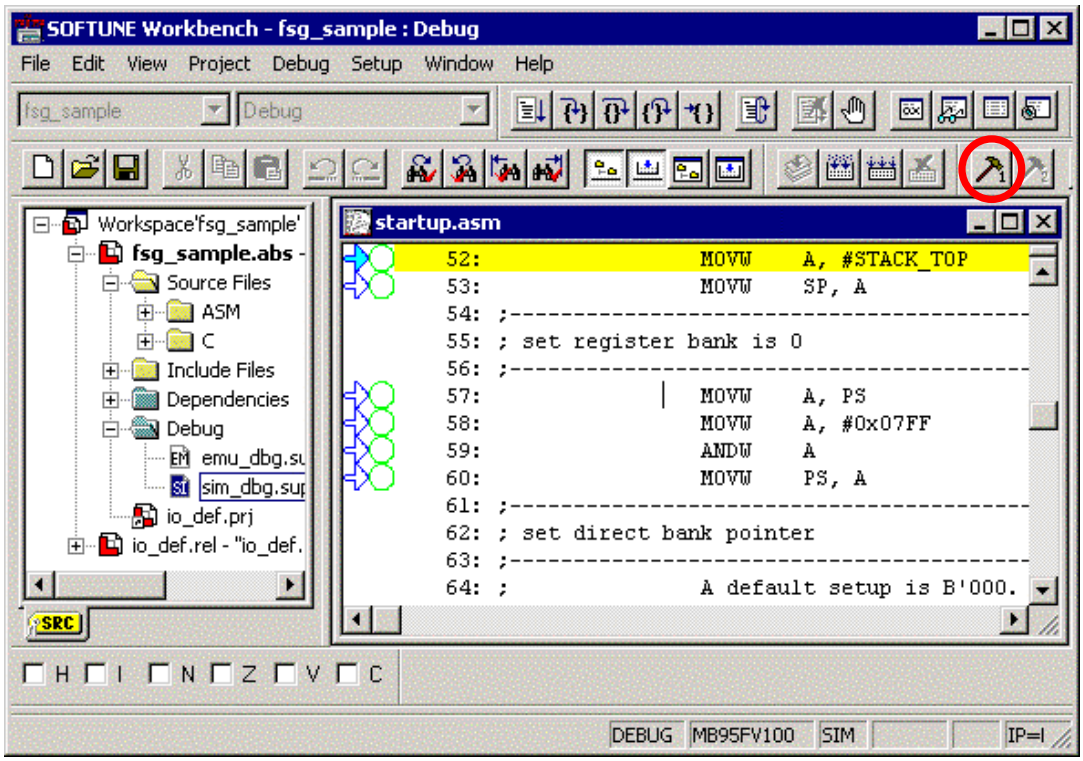
(3) Select the Number, File Kind, and Entry (batch file) to be input in the [Customize Bar Setting] dialog.

 can be used to refer to the batch file.

This example enters the batch file (**`fsg_sample.prc`**) made in Tutorial in ***Chapter 4*** (**`C:\Softune\sample\fsg\8fx_sample\fsg_sample.prc`**).



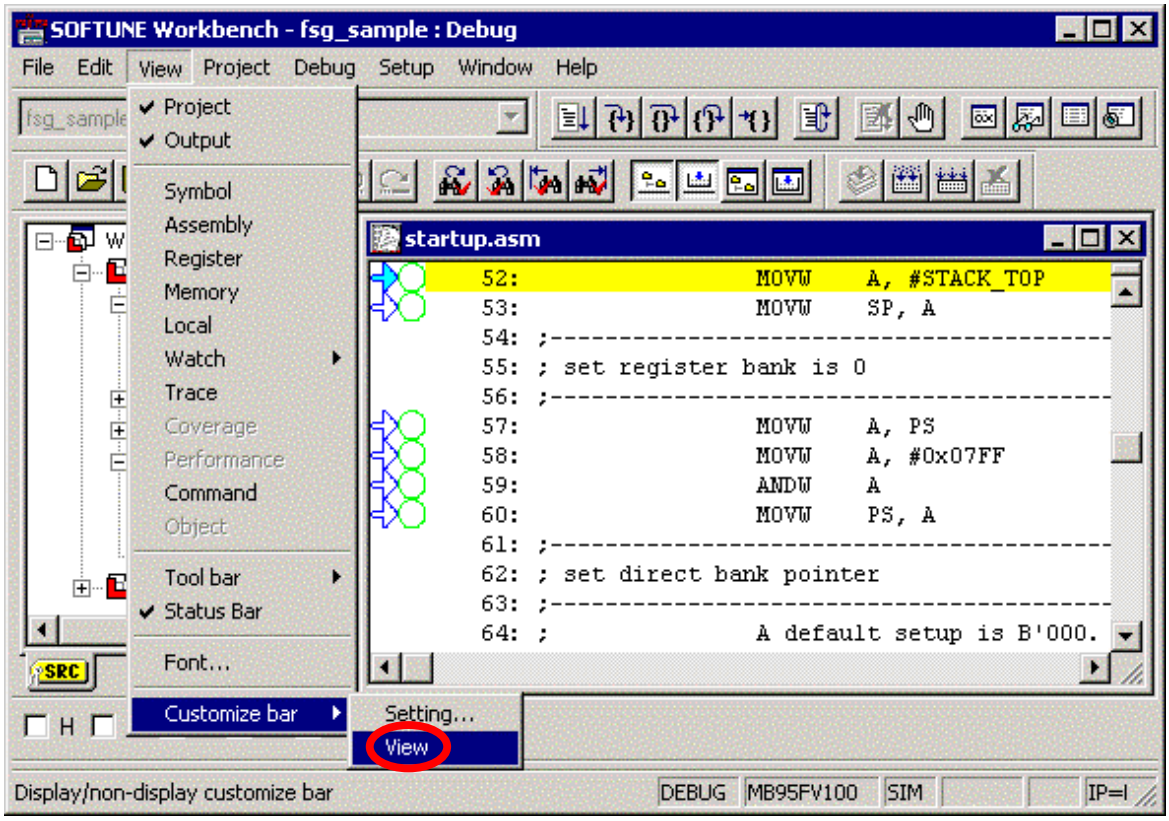
(4) The customize bar is entered and displayed on the window.



One Point !

= Displaying customize bar =

Select [Customize bar]-[View] from the [View] menu.



One Point !

= Debug mode =

The batch file can be input and executed only in the debug mode.

The debug mode is set when the following items are enclosed in the dotted lines.

- SIM = Simulator Debugger
- EML = Emulator Debugger

5.4 Setting Customize Build

■ Merit in Setting Customize Build

When executing Compile, Assemble, Make, and Build, the user can register another tool before and after executing the respective tools.

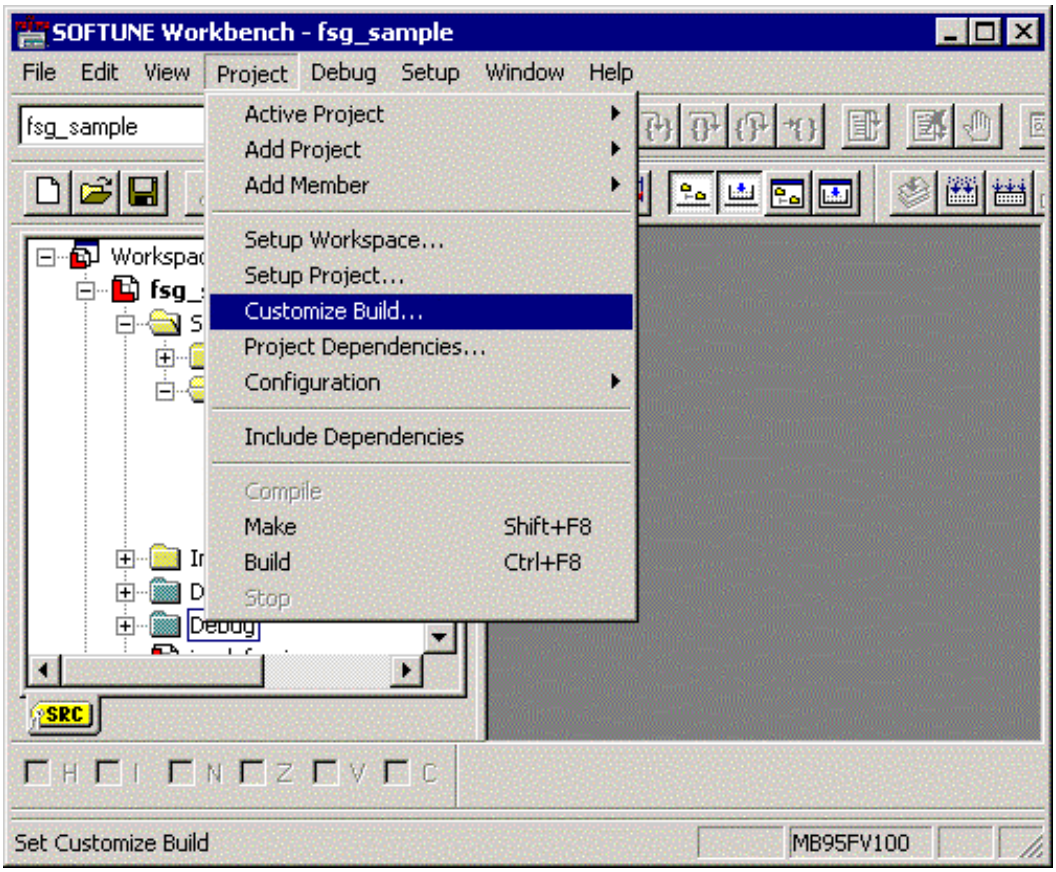
The following example registers the object module conversion tool (S format Adjuster) after starting the Converter.

This setting is saved every project.

■ Setting Procedure for Customize Build

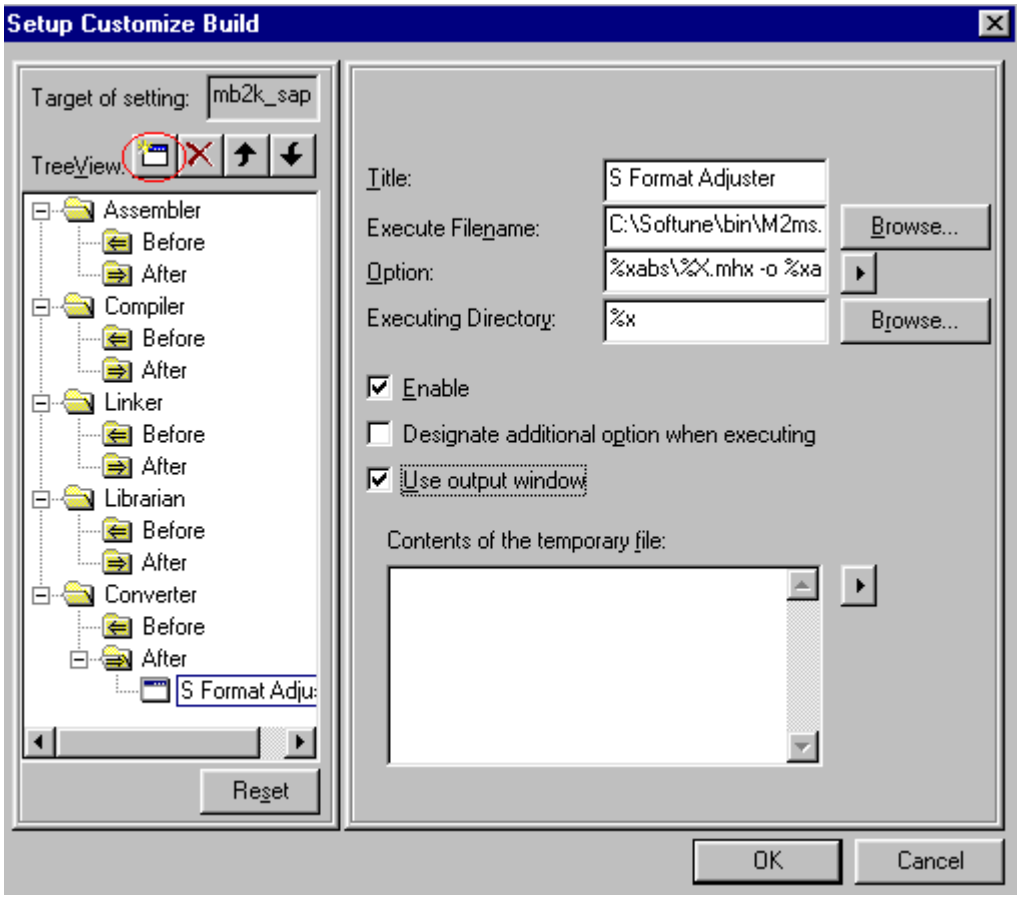
《 Setting procedure 》

(1) Select [Customize Build...] from the [Project] menu.



(2) Select “Converter-After” in the [Setup Customize Build] dialog and click the New button in Tree View to set the following items.
Macro define can be used to for the option.
In this example, a file is made in the ABS folder in the Project directory where the target file is created.

- Title → S Format Adjuster
- Execute Filename → **`C:\Softune\bin\M2ms.exe`**
- Option → %xmask\abs\%X.mhx -o %xmask\abs\%X.ahx -ran 0x4000, 0xFFFF –S1
- Executing Directory → %x



(3) Click the [OK] button to complete the setting.

After the target file is made, the Converter will start and the S format Adjuster will also start.

APPENDIX

Appendix Q&A

A FAQs for Project Design

- Q1 How do I move the *SOFTUNE* Workbench project to another machine or drive?
- Q2 Why is not the source file displayed when debugging is performed by a machine that did not execute Build?
- Q3 How do I make projects with different language tool options and Member configurations of source files?
- Q4 What is the difference between project types (loadmodule, relocatable, and library)?
- Q5 What does information specified in Target MCU in the [Create] dialog reflect?
- Q6 What is information set in the [Setup CPU Information] dialog used for?
- Q7 Is the extension determined by the type of file?
- Q8 How do I change the linkage order?
- Q9 Is the dependent source file at Make execution compiled when only the include file is changed?

B FAQs for Debugger

- Q1 What should I do if the Debugger cannot be started?
- Q2 When the conversion is not performed normally after symbolic addressing is performed.
- Q3 How to set the path count of a breakpoint during using the Emulator Debugger
- Q4 Why are the settings of breakpoints are changed when the target file is changed?
- Q5 How to save trace results and evaluation results such as memory dump to a file
- Q6 Does the Emulator support coverage measurement?
- Q7 Does the Emulator support time measurement?
- Q8 Does the Emulator support event setting?
- Q9 Does the Emulator support the Debugger memory map?
- Q10 Does the Emulator support the monitoring function?
- Q11 Are there any restrictions on the instruction count between branches that can be displayed by trace?

- Q12 What should I do if the progress bar showing “Processing” is displayed and the Debugger is not in the “execute” state when the program is executed by the Emulator?
- Q13 How to shorten the flash memory update time?
- Q14 What does the Debugger do if the user suspends the operation during executing in the standby mode by the Emulator?

C Others

- Q1 How to check the version of ***SOFTUNE*** Workbench and each language tool.
- Q2 How to change the size of the font displayed in each ***SOFTUNE*** Workbench window.
- Q3 How to retrieve a file displayed in the ***SOFTUNE*** Workbench Project window.
- Q4 How to check details of the sample I/O register file.
- Q5 How to change the record length of Motorola S format data.
- Q6 How to get the notices on using the Assembler in C.
- Q7 How to read projects developed by V30L26 or earlier ***SOFTUNE*** Workbench by V30L27 or later ***SOFTUNE*** Workbench?

A FAQs for Project Design

Q1 How do I move the *SOFTUNE* Workbench project to another machine or drive?

Answer

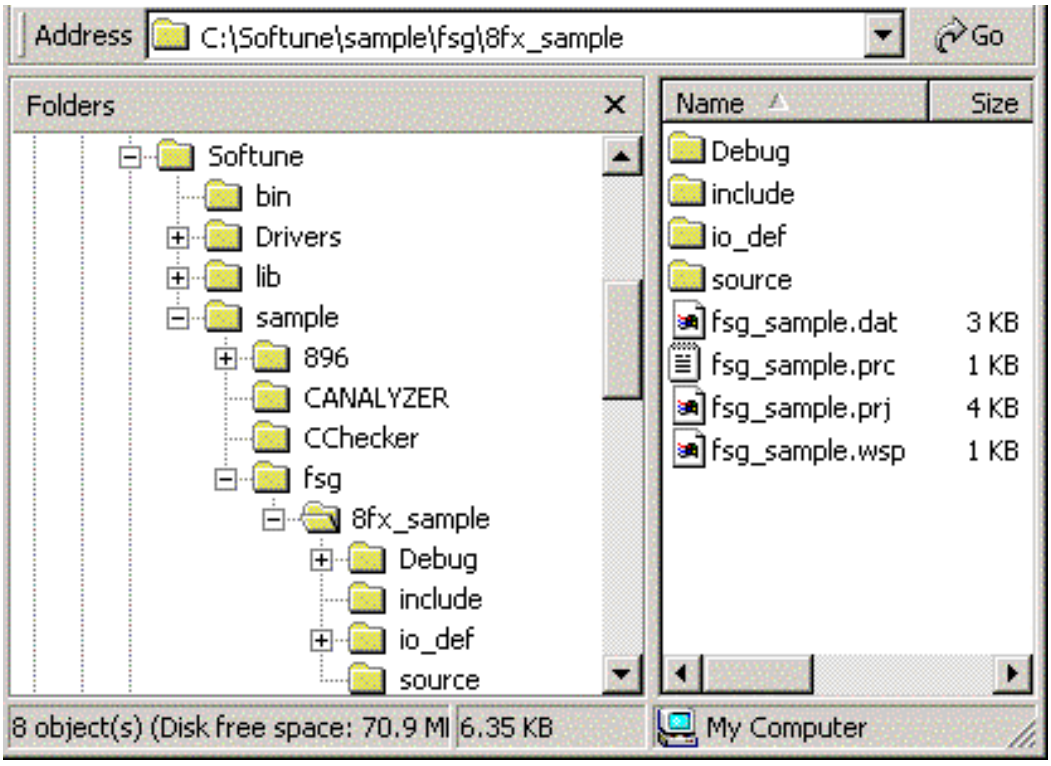
SOFTUNE Workbench manages project information stored in the workspace using the workspace file (workspace name.**wsp**). It also manages Member information and information on various language tool options stored in the project using the project file (project name.**prj**) and the option data file (project name.**dat**).

If project information managed using the workspace file and Member information managed using the project file are on the same drive as the workspace file and project file, ***SOFTUNE*** Workbench manages these information with the relative path; if they are on a different drive from the workspace file and project file, ***SOFTUNE*** Workbench manages with the absolute path.

If there are no changes in relative path information, ***SOFTUNE*** Workbench will automatically recognize path information even if the project is moved. However, if the information is managed by the absolute path, it must be reset when the drive name or path information is changed.

Setting all necessary files under the workspace file and project file will be helpful when moving the project.

Fujitsu recommends the user should make the workspace/project configuration as described in ***Chapter 4*** Tutorial.

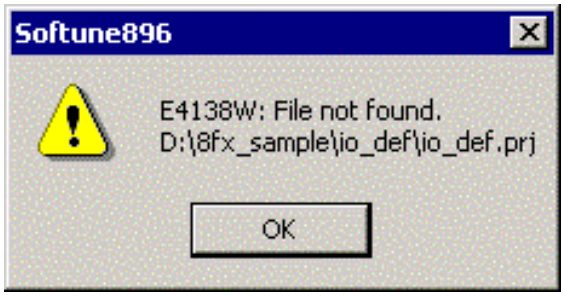


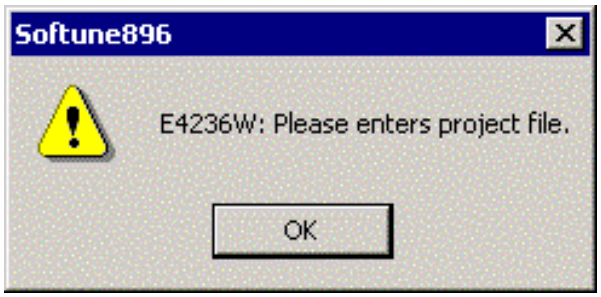
If the user moves the project to another machine or drive, check the following items and change each setting as necessary.

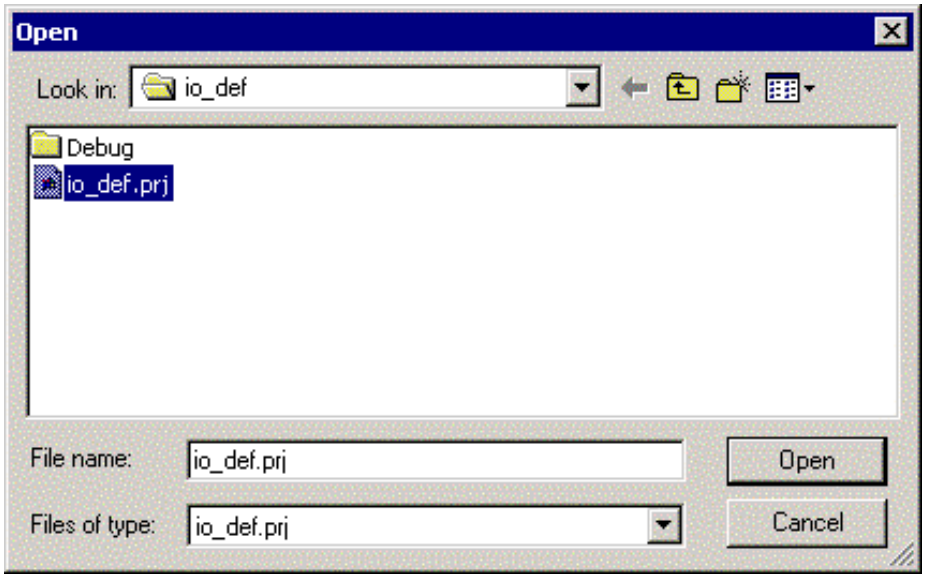
《 Setting procedure 》

(1) Use ***SOFTUNE*** Workbench to open the moved workspace and check whether the input project and Member file are in the workspace. If not, they must be reset.

If there is no project, the following message is displayed. Specify the path of the project and input the project in the workspace.

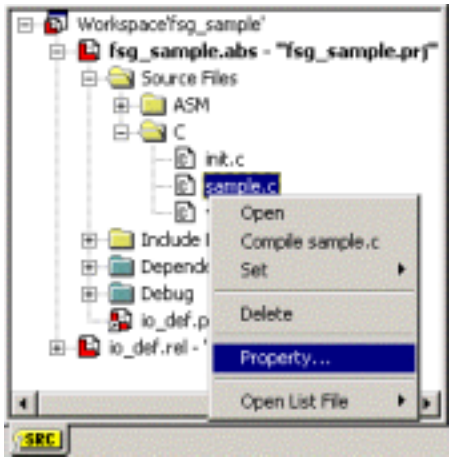


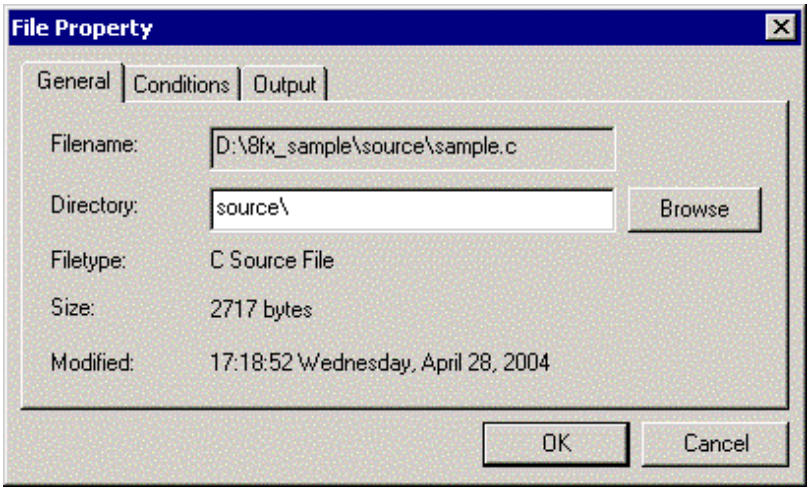




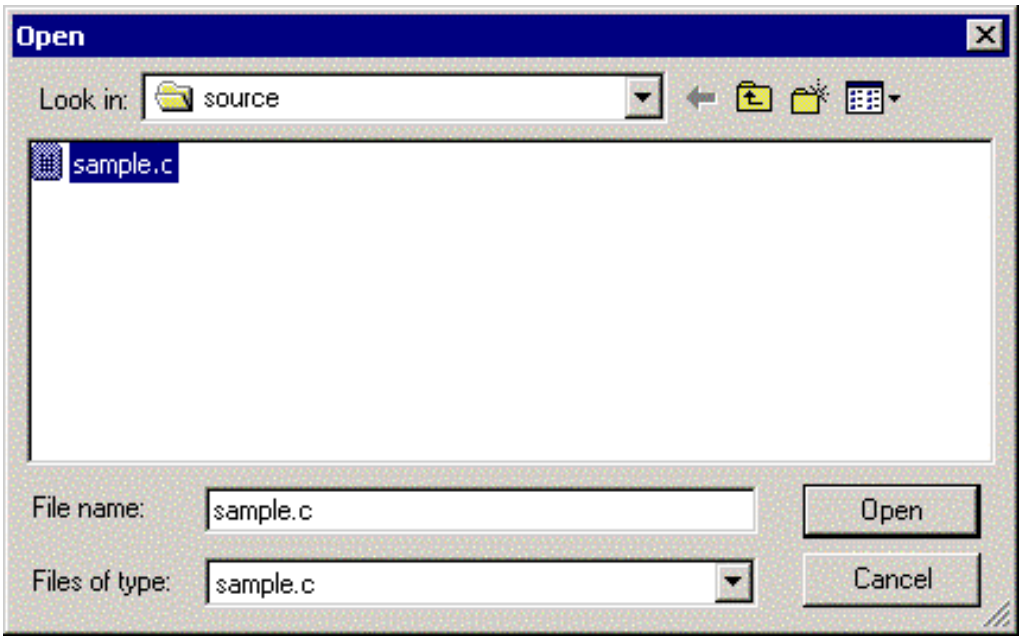
If there is no Member file, it can be reset by changing its properties.

Click the file and select [Property...] from the shortcut menu to open the “File Property” dialog.





Click the [Browse] button to set the path of the Member file in the [Open] dialog.



(2) Check the include paths of the C Compiler and Assembler and the library search path of the Linker.

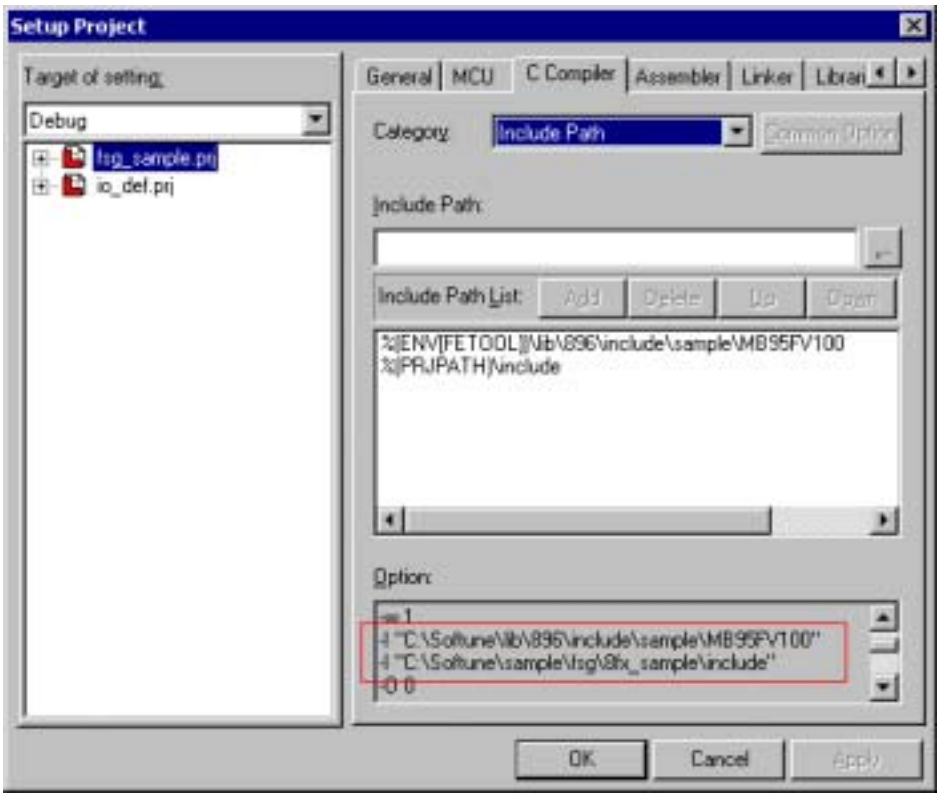
Check the include path in the [C Compiler] and [Assembler] tabs and the library search path in the [Linker] tab in the [Setup] dialog, and change them as necessary.

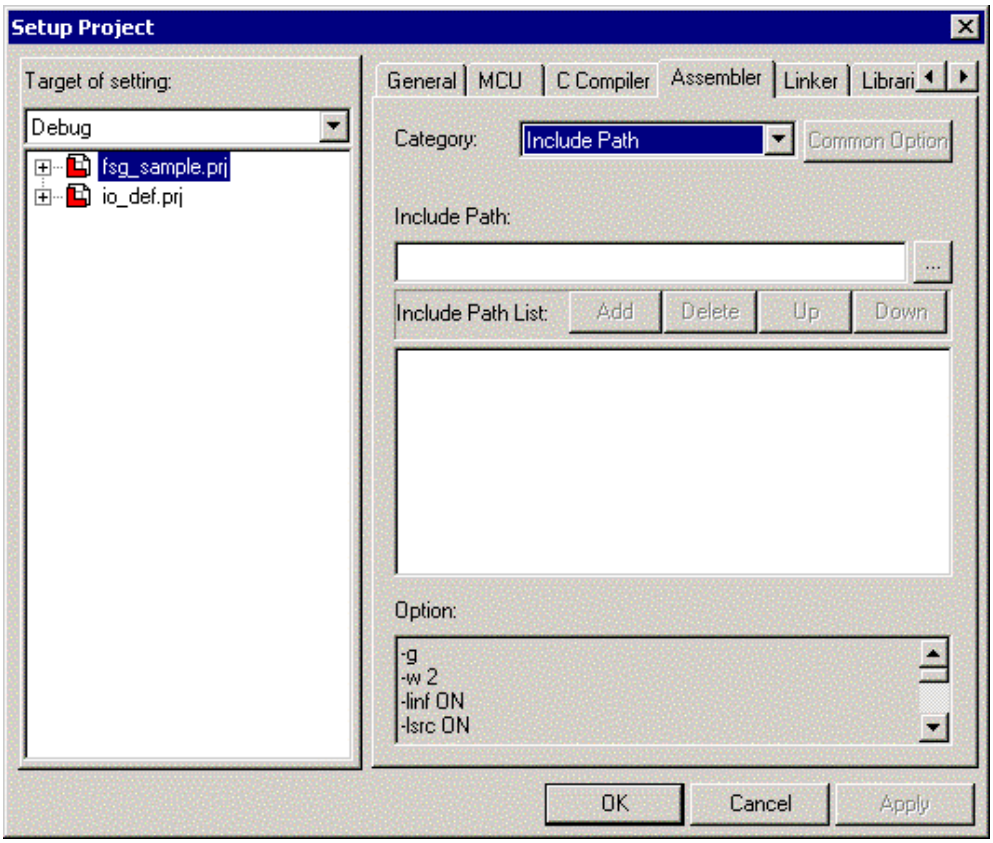
The macro may be used to set the include path.

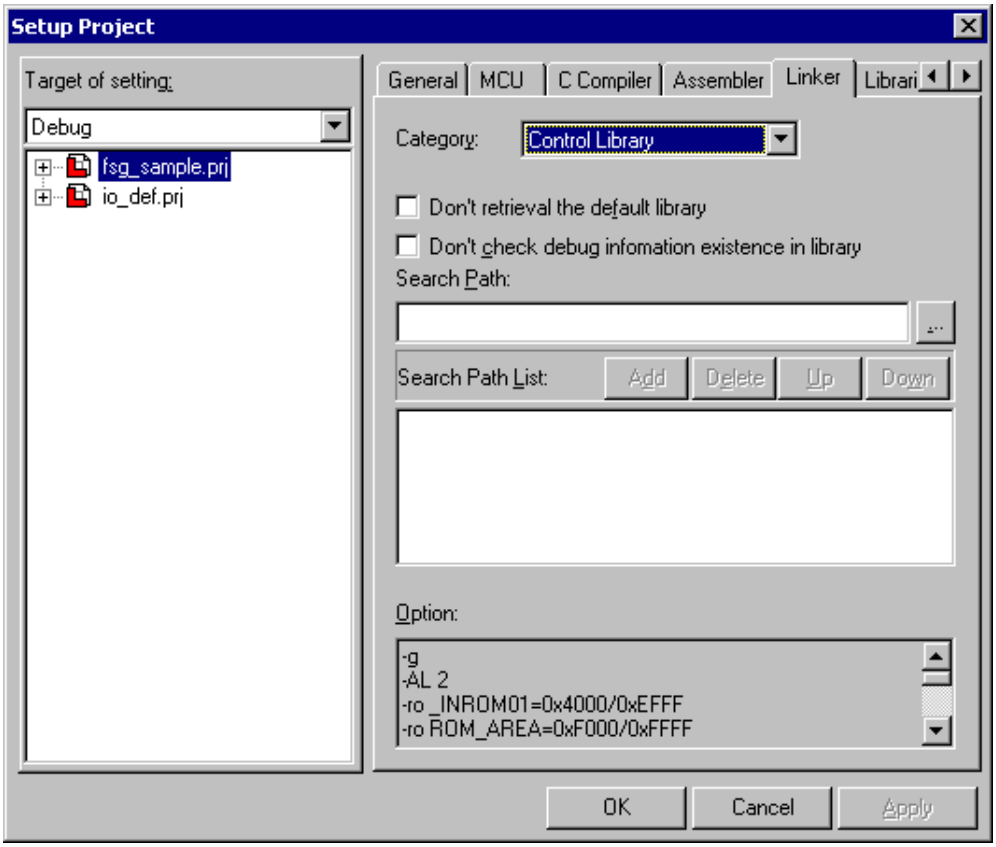
An example of setting the include path using the macro is shown below.

The description using the macro is displayed in the Option column (enclosed in dotted lines) and can be used to check an expanding path.

- %(ENV[FETOOL])\lib\896\include\sample\MB95FV100 → ***SOFTUNE*** install directory
- %(PRJPATH)\include → Project directory



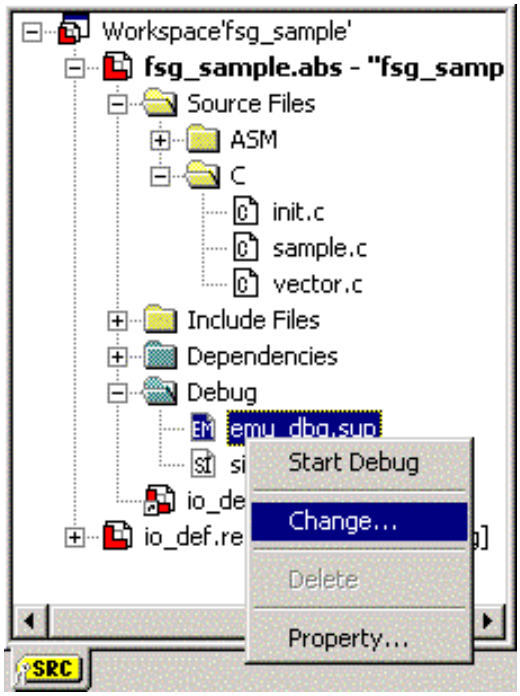

Setup Project
Target of setting:
Debug
fsg_sample.prj
io_def.prj
General
MCU
C Compiler
Assembler
Linker
Librari
Category:
Include Path
Common Option
Include Path:
Include Path List:
Add
Delete
Up
Down
Option:
-g
-w 2
-linf ON
-lsrc ON
OK
Cancel
Apply

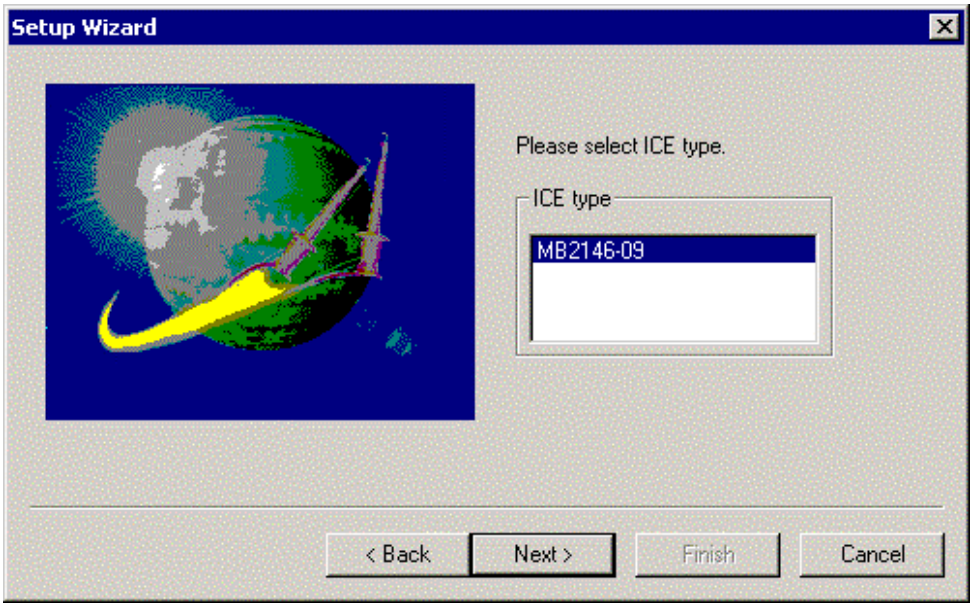

Setup Project
Target of setting:
Debug
fsg_sample.prj
io_def.prj
General
MCU
C Compiler
Assembler
Linker
Librari
Category:
Control Library
Don't retrieval the default library
Don't check debug infomation existence in library
Search Path:
Search Path List:
Add
Delete
Up
Down
Option:
-g
-AL 2
-ro _INROM01=0x4000/0xEFFF
-ro ROM_AREA=0xF000/0xFFFF
OK
Cancel
Apply

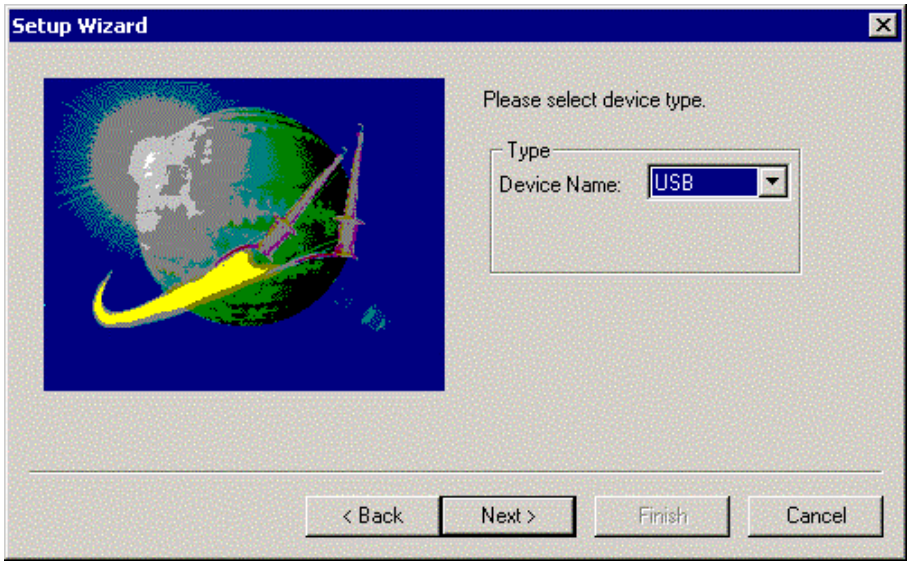
(3) Execute [Make] or [Build] in the [Project] menu to make a target file. Check that no error has occurred.

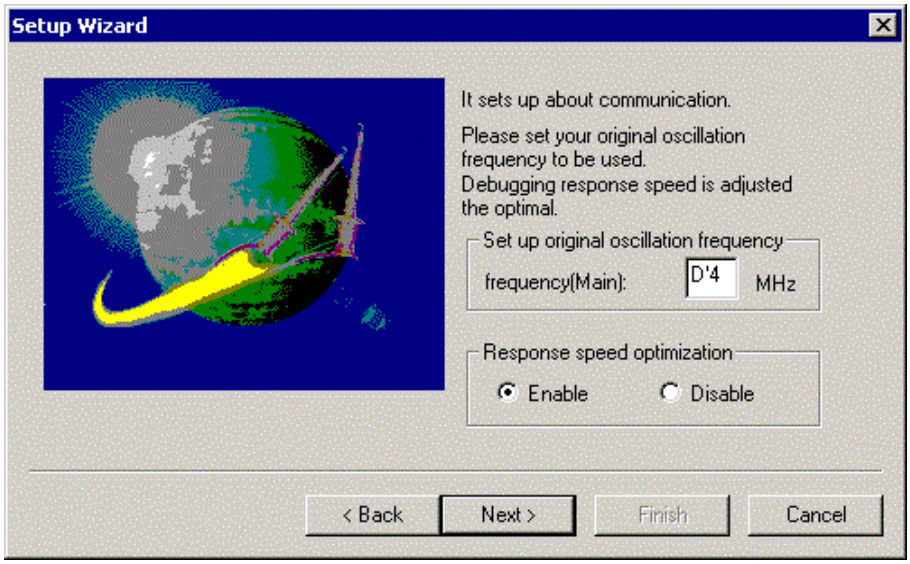
(4) Check Debugger setup information before starting debugging.

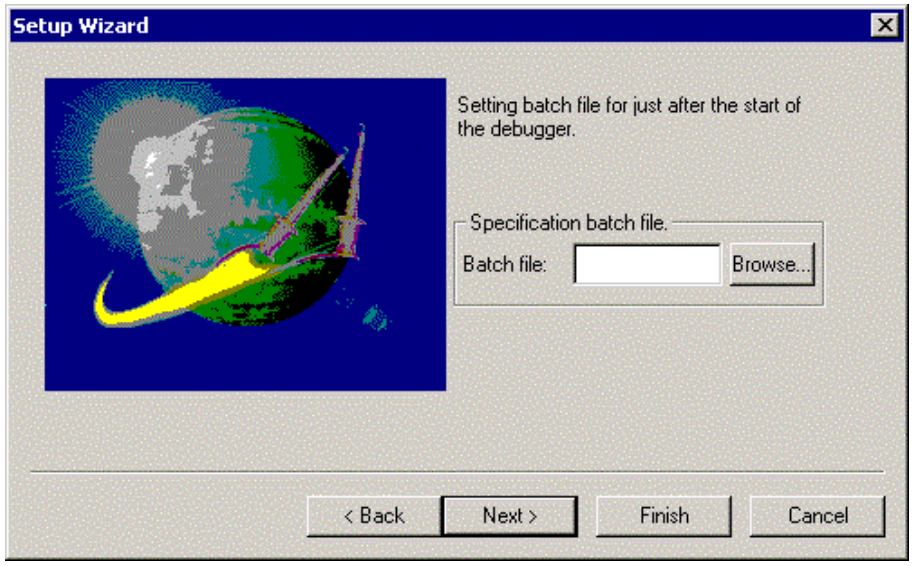
Click the setup file, select [Change...] from the shortcut menu to start the setup wizard, and check the communication device setting (for the Emulator Debugger) and the batch file setting before and after loading.

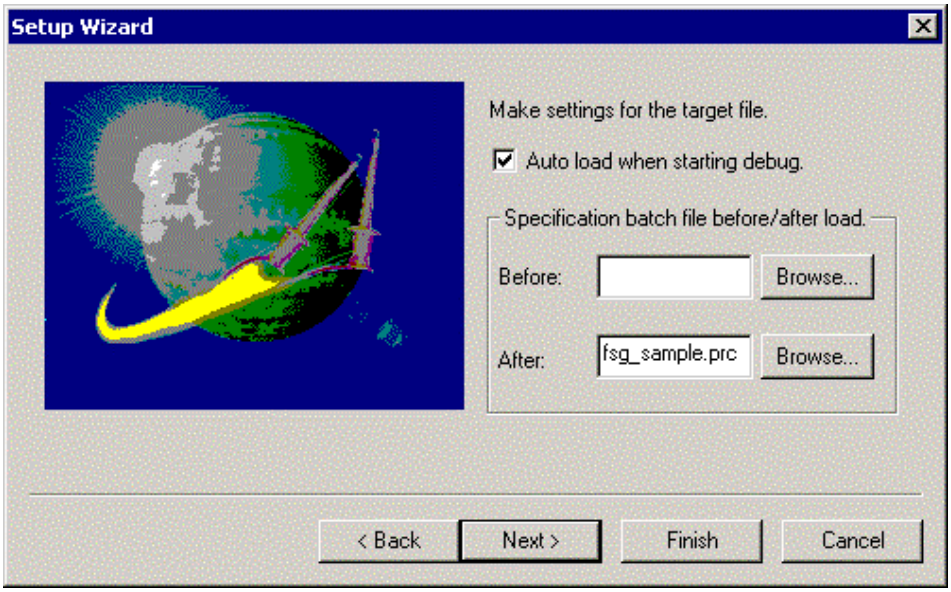


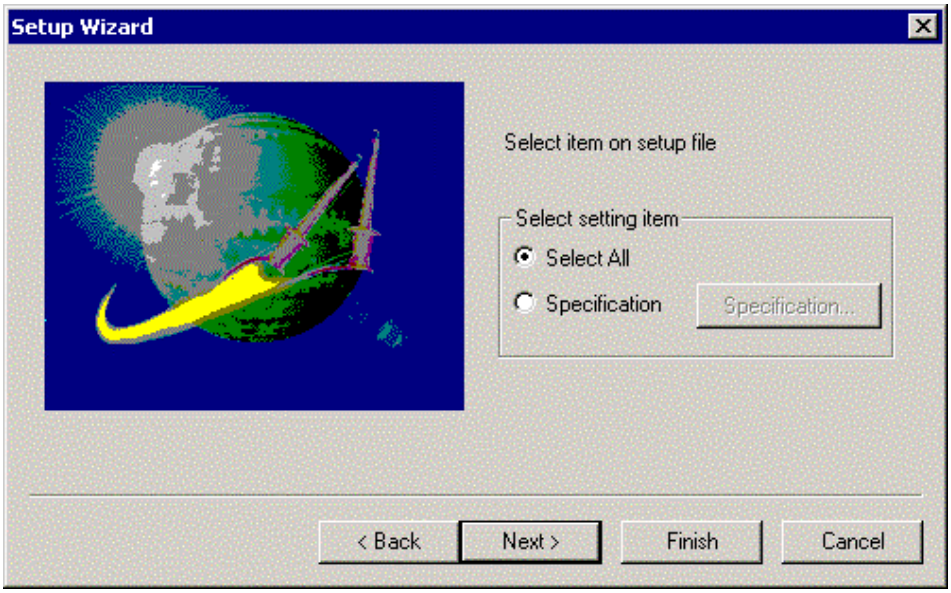


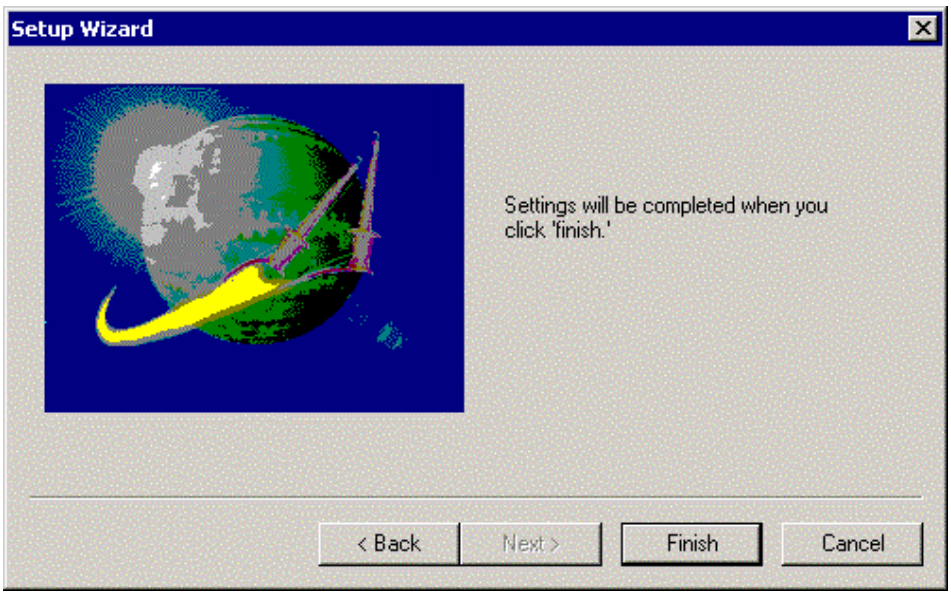

Setup Wizard
Please select device type.
Type
Device Name:
USB
< Back
Next >
Finish
Cancel


Setup Wizard
It sets up about communication.
Please set your original oscillation frequency to be used.
Debugging response speed is adjusted the optimal.
Set up original oscillation frequency
frequency(Main):
D'4
MHz
Response speed optimization
Enable
Disable
< Back
Next >
Finish
Cancel


Setup Wizard
Setting batch file for just after the start of the debugger.
Specification batch file.
Batch file:
Browse...
< Back
Next >
Finish
Cancel


Setup Wizard
Make settings for the target file.
Auto load when starting debug.
Specification batch file before/after load.
Before:
Browse...
After:
fsg_sample.prc
Browse...
< Back
Next >
Finish
Cancel

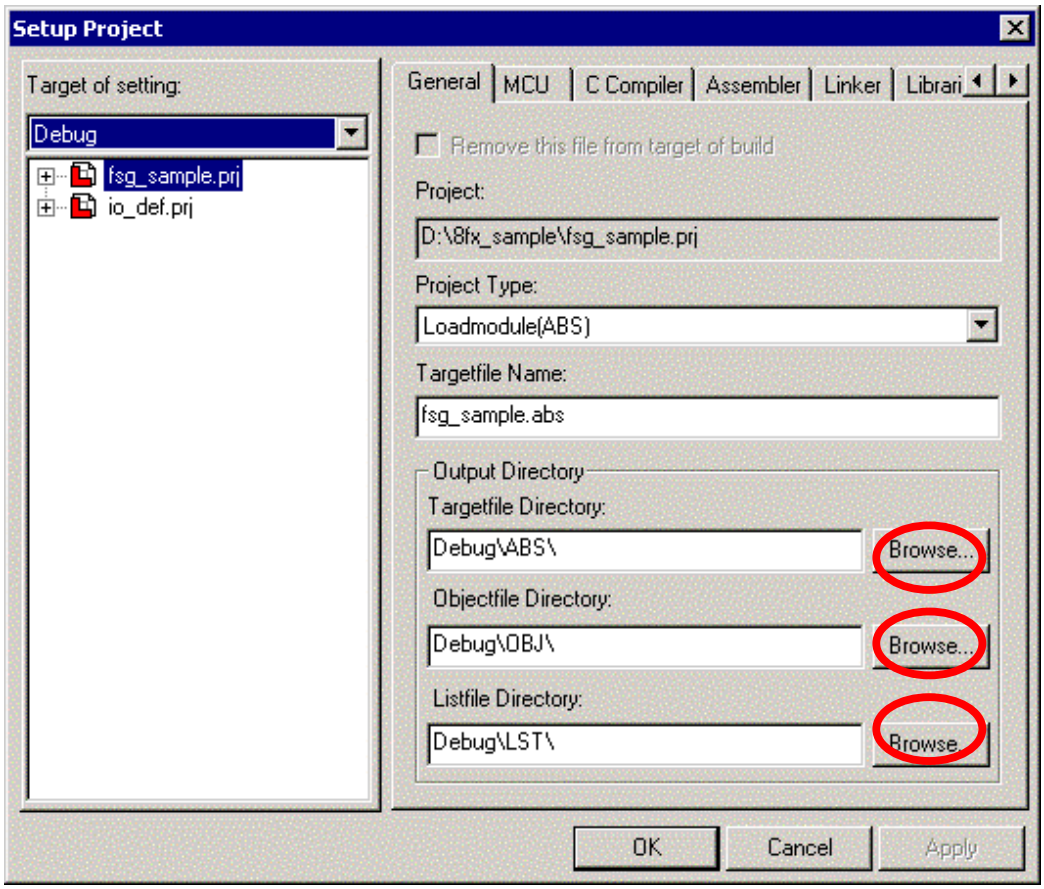

Setup Wizard
Select item on setup file
Select setting item
Select All
Specification
Specification...
< Back
Next >
Finish
Cancel


Setup Wizard
Settings will be completed when you click 'finish.'
< Back
Next >
Finish
Cancel

One Point !

= Setting in [General] tab in [Setup] dialog =

The [Browse...] button can be used to change the paths for the target directory, object file directory, and list file directory. However, if the directory is specified with the absolute path resetting is necessary when the project is moved. It is best to perform operation at the default settings.



Q2 Why is not the source file displayed when debugging is performed by a machine that did not execute Build?

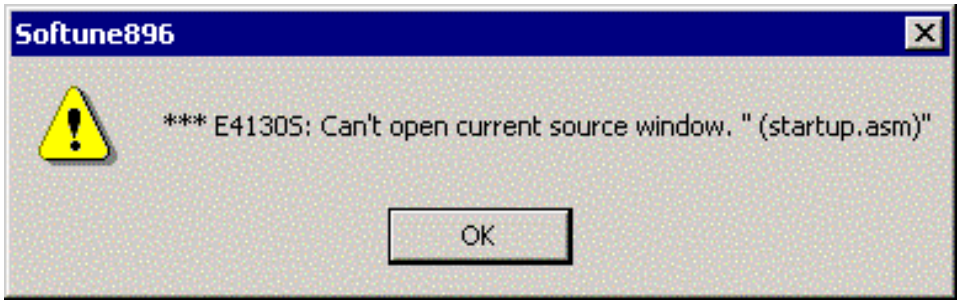
Answer

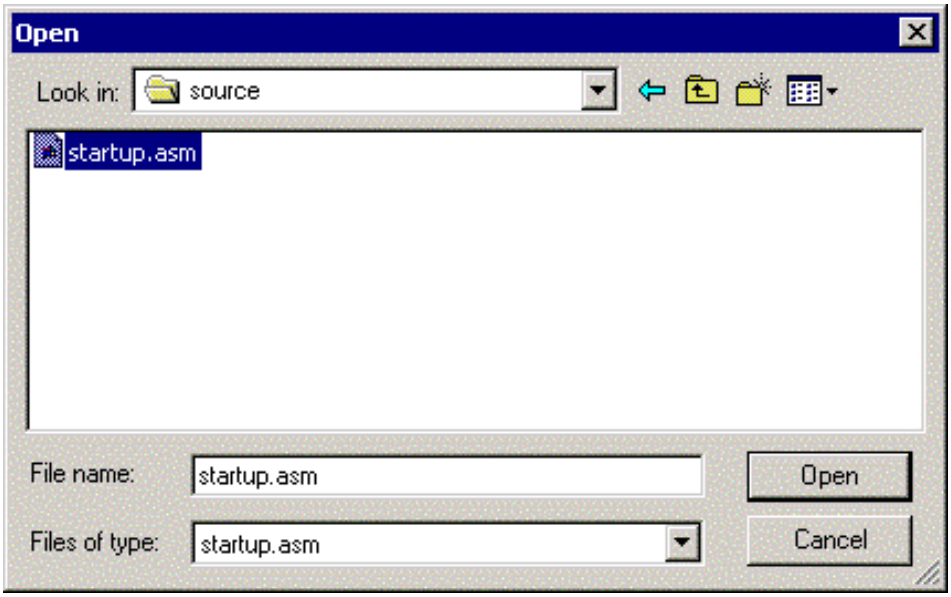
Because the target file has a path information of the source file when Make or Build is executed, if the project directory is moved or the project is moved from another machine, the source file cannot be displayed properly when the Debugger is started.
To display the source file properly, set the following items.

《 Setting procedure 》

(1) When [Start debug] is selected from the [Debug] menu to start the Debugger, the following dialog is opened.

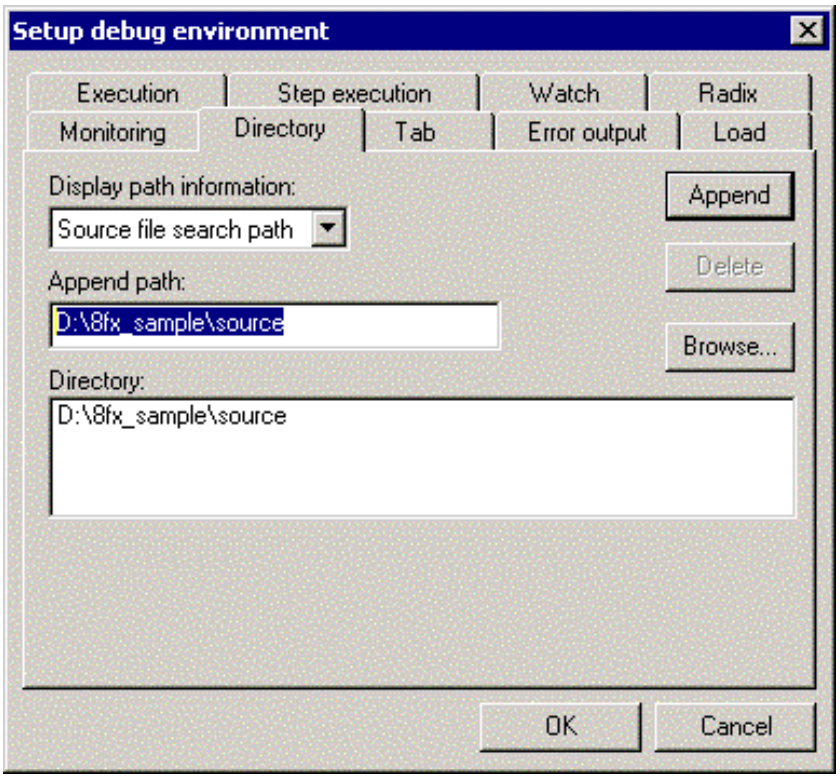
Follow the dialog to specify the directory containing the source file.





(2) Select [Debug environment]-[Debug environment...] from the [Setup] menu to open the [Setup debug environment] dialog.

Set all directories containing the source file in the [Directory] tab.



Q3 How do I make projects with different language tool options and Member configurations of source files?

Answer

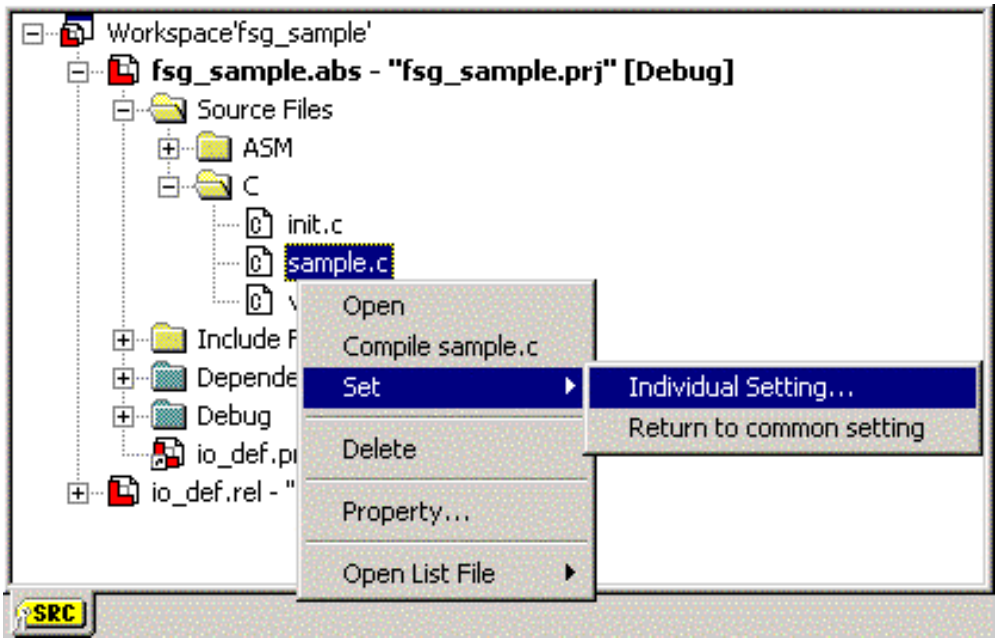
By adding project configurations, the user can make project setting with different setting conditions for language tool options and Member files for testing and the optimization trial.

For details, see ***Section 4.4.1*** Adding Project Configuration.

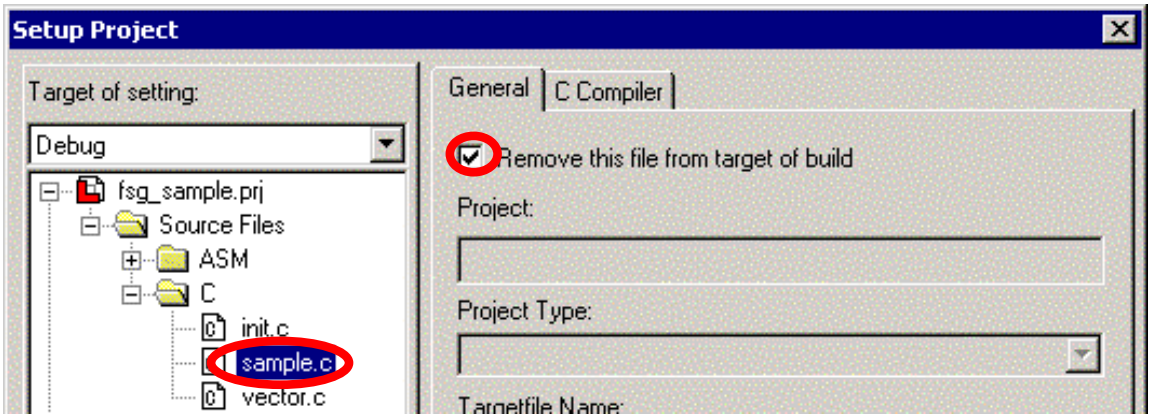
The procedure for removing one file from the Build operation is shown below.

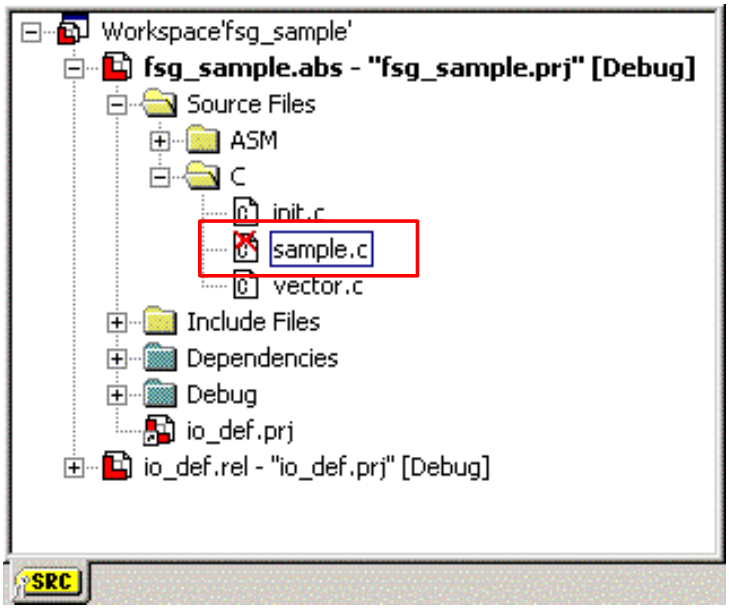
《 Setting procedure 》

(1) Click the source file to be removed from the Build operation and select [Set]-[Individual Setting...] from the shortcut menu.



(2) Check the “Remove this file from target of build” checkbox in the [General] tab in the [Setup Project] dialog.





One Point !

= Option file in OPT folder =

SOFTUNE Workbench makes a temporary file required for compiling/assembling in the **0PT** folder.

Note that this is a temporary file, so its correction will not affect ***SOFTUNE*** Workbench.

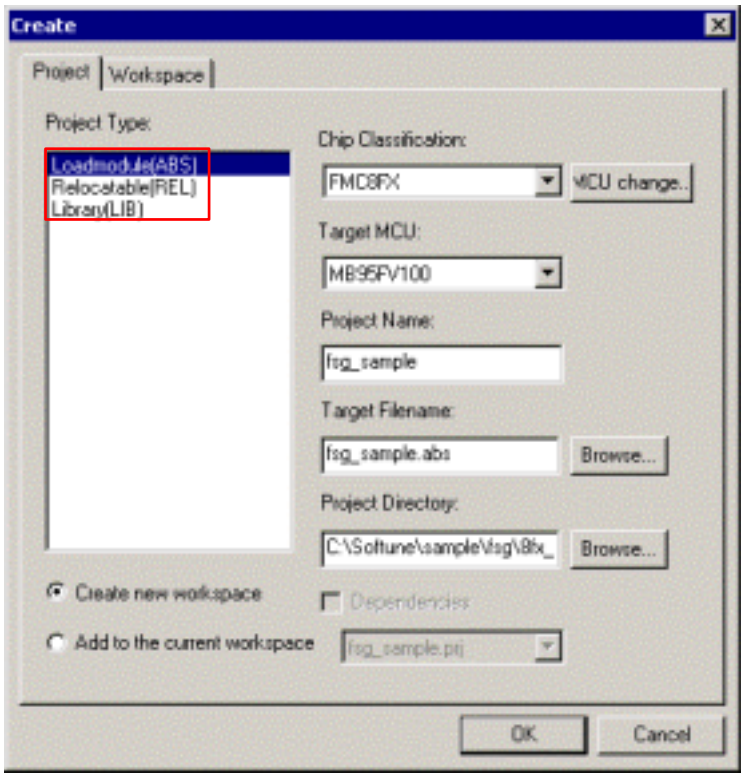
Q4 What is the difference between project types (loadmodule, relocatable, and library)?

Answer

There are three types of projects that are set in the [Create] dialog: Loadmodule (ABS), Relocatable (REL), and Library (LIB).

Select the Loadmodule (ABS) for a project to make a target fie. The Loadmodule (ABS) determines the absolute address for all modules.

The features of the Relocatable (REL) and project type of the Library (LIB) are described below.



- Relocatable (REL)

  The project type of this relocatable format links multiple object files into one relocatable format (REL) file.

  Only a part of the information in the relocatable file cannot be read.

  The time required for link processing is shorter than that for the library file.

- Library (LIB)

  The project type of the library links multiple object files into one library (LIB) file.

  Unlike the relocatable file, the multiple object information in the library file is managed individually; it can be read as required or new information be input to the library.

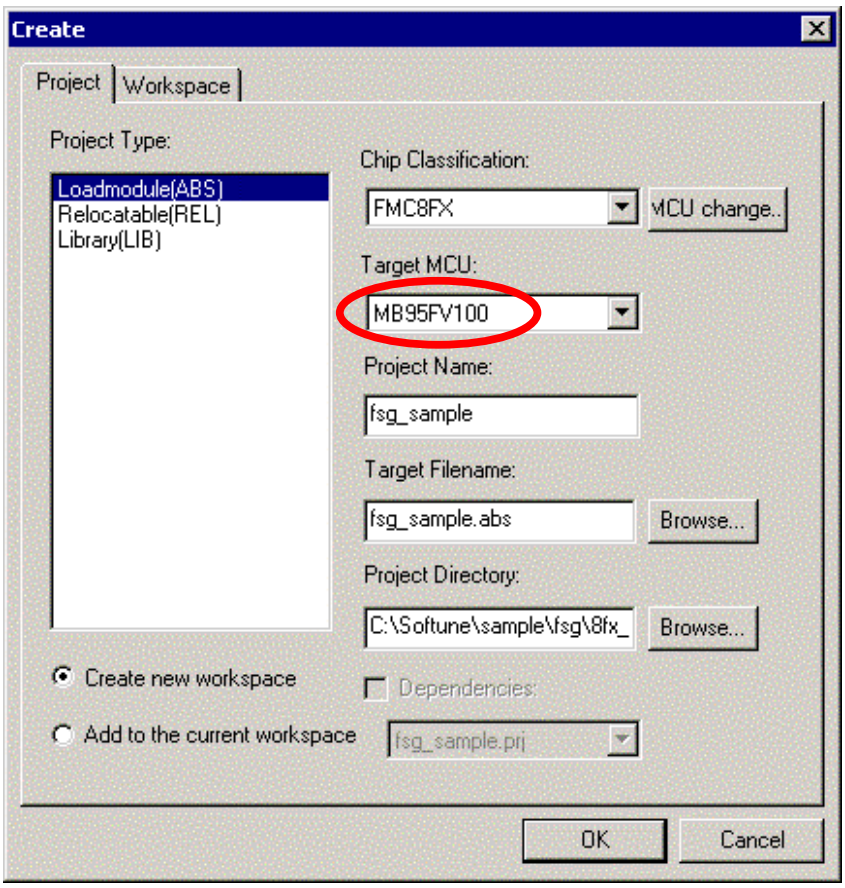
  The time required for link processing is longer than that for the relocatable file.

Q5 What does information specified in Target MCU in the [Create] dialog reflect?

Answer

The settings in Target MCU are used to set the following information:

- Making of I/O area module using sample I/O register file
- CPU Options (including check for section arrangement) for Compiler, Assembler, and Linker
- Default settings when [Set CPU Information...] in [MCU] tab in [Setup Project] dialog (See ***Q6*** for details.)
- Memory map information when Debugger started



Q6 What is information set in the [Setup CPU Information] dialog used for?

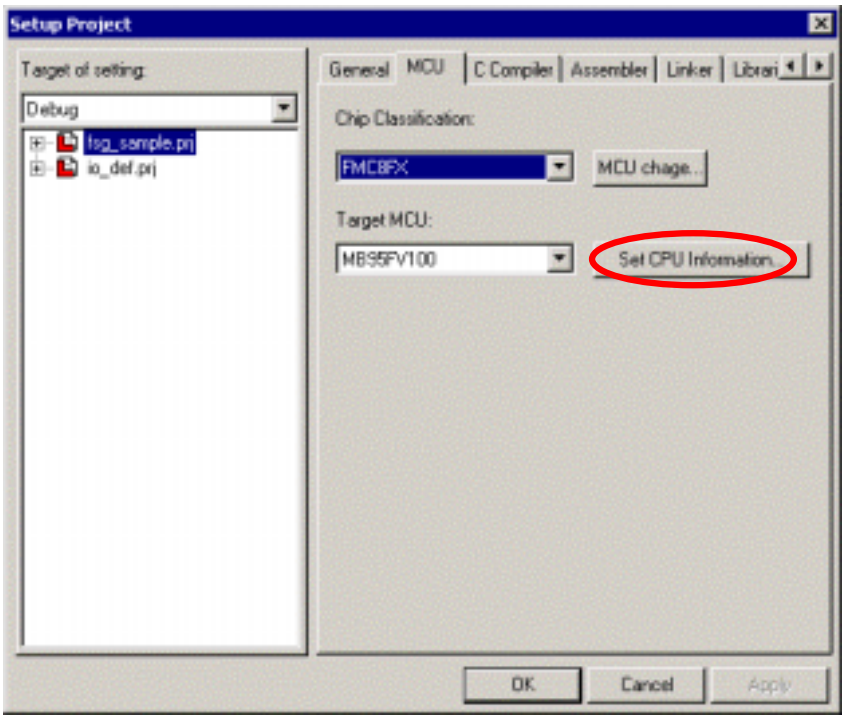
Answer

Setting information in the [Setup CPU Information] dialog is used for the Simulator Debugger or the Emulator Debugger.

Click [Set CPU Information...] in the [MCU] tab in the [Setup Project] dialog to open the [Setup CPU Information] dialog.

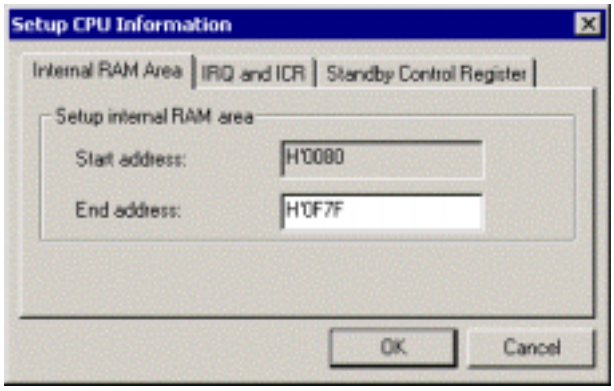
The settings are explained assuming that the target CPU is the MB95FV100.

When the MB number of the target MCU is selected, the recommended default is set automatically.



- [Internal RAM Area] tab

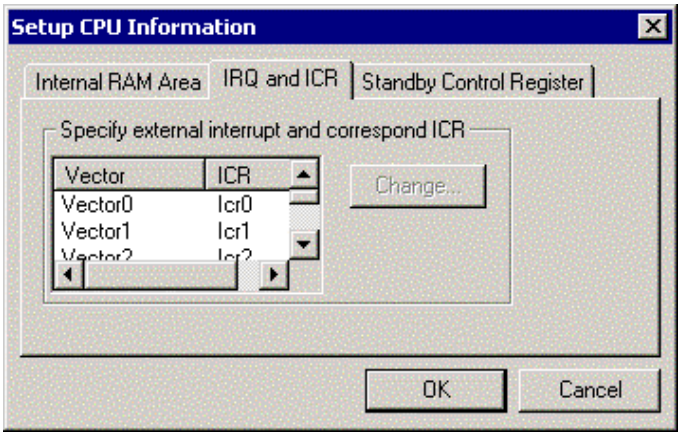
  Specify the start and end addresses of the internal RAM area



- [IRQ and ICR] tab

  Set the Simulator Debugger.

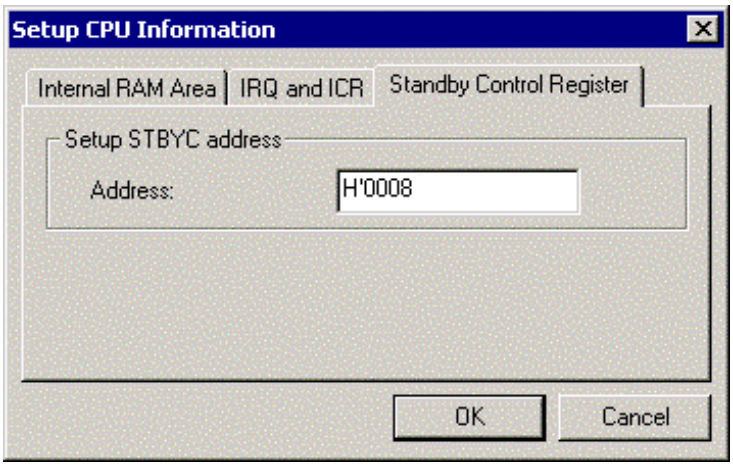
  Specify the external interrupts and correspond ICRs.



- [Standby Control Register] tab

  Set the Simulator Debugger.

  Specify the address of the standby control register.



Q7 Is the extension determined by the type of file?

Answer

The extension is determined by default. However, any file with any extension can be recognized as a target file.

When adding a Member to the project, any file with any extension can be recognized as a target file by changing the member type.

For example, when adding an Assembler file to the project, the file extension is usually "~**.asm**." However, a file with the extension "~**.s**" can be recognized as an Assembler file.

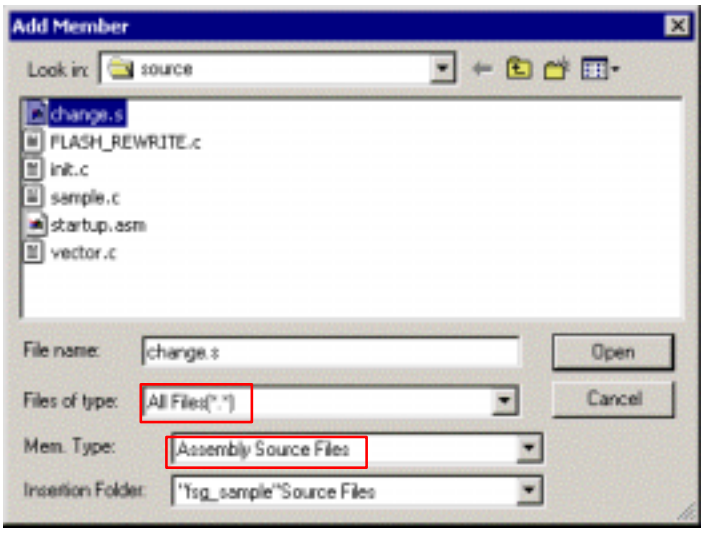
The setting procedure is described below.

One Point !

(1) Select [Add Member...]-[File] from the [Project] menu to open the [Add Member] dialog.

(2) Select "All Files (*.*)" for Files of Type and "Assembly Source Files" for Mem. Type.

Specify the file name and click the [Open] button.



The above example recognizes the added "**change.s**" as an Assembly source file.

Similarly, when any of C source Files, Object Files, Library Files, or Relocatable Files is set for Mem. Type, each file is recognized as a target file, irrespective of the extension.

= File types and extensions by default =

Mem. Type is set to “Auto” by default.

The recognizable file types and extensions in this mode are listed below:

- C Source files
  ~.**c**, ~.**i**
- Assembly source files
  ~.**asm**
- Object files
  ~.**obj**
- Relocatable files
  ~.**rel**
- Library files
  ~.**lib**
- Include files
  ~.**h**
- Other files
  All files with other extensions are recognized as other files.

Q8 How do I change the linkage order?

Answer

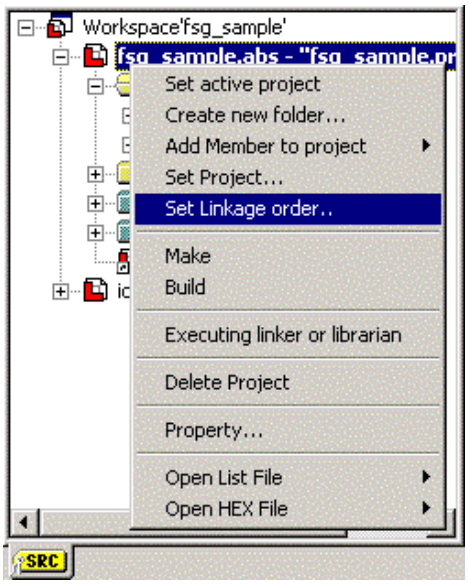
The user can change the linkage order in the Project window.

The order before being changed is that in which source files are added to the Member.

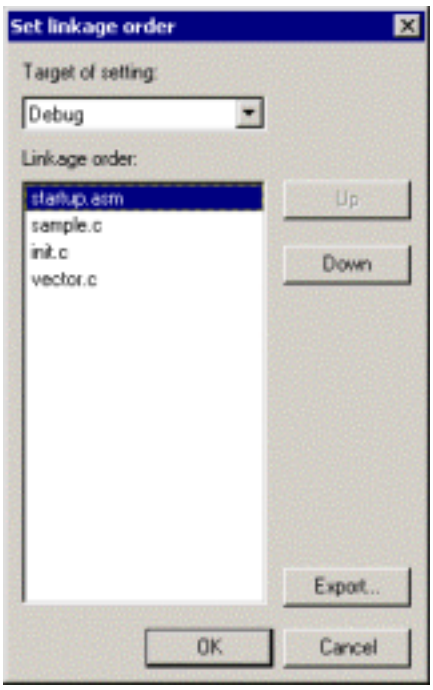
The setting procedure is described below.

《 Setting procedure 》

(1) Specify the Project target file name and select [Set Linkage order...] from the shortcut menu.



(2) In the [Set Linkage order] dialog, rearrange the items from top in the order in which the user wants to link. Use the [Up] and [Down] buttons to change the order. After completing the settings, click the [OK] button.



Q9 Is the dependent source file at Make execution compiled when only the include file is changed?

Answer

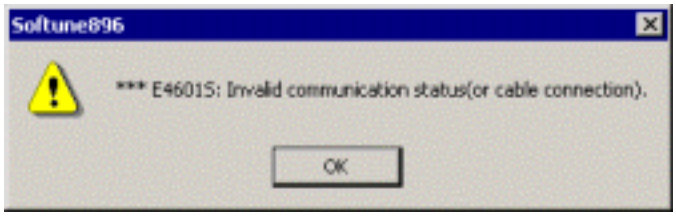
If the include file is changed even when there is no change in the source file, all the dependent files are compiled by executing Make.

B FAQs for Debugger

Q1 What should I do if the Debugger cannot be started?

Answer

- If the POWER LED of the Emulator (MB2146-09) is green, power may not be supplied to the target board or the target board may not be connected correctly to the MB2146-09. Check the supply voltage level and the connection to the MB2146-09. If the POWER LED is red, the MB2146-09 may not be connected correctly to the host. Check the connection to the host. The POWER LED is orange when the connection is normal.
- If the user gets the following error message when the POWER LED of the MB2146-09 is orange, disconnect the USB cable from the PC temporarily, reconnect it to the PC, and then start the Debugger. (If this error occurs during debugging, stop debugging temporarily and disconnect and reconnect the cable to recover.)



- If the communication error still occurs after the above operation, the USB driver for the MB2146-09 may not be installed. For details about installation and connection, refer to:
  - Operation Manual Appendix D Setting USB Interface
  - Manual attached to the MB2146-09

Q2 When the conversion is not performed normally after symbolic addressing is performed.

Answer

Symbolic addressing is available for line assembling, event setting, and jumping function in the Memory window.

Use symbol name to specify address symbols such as arrays.

However, for value symbols, such as array elements, use the symbol name with an address operator (&).

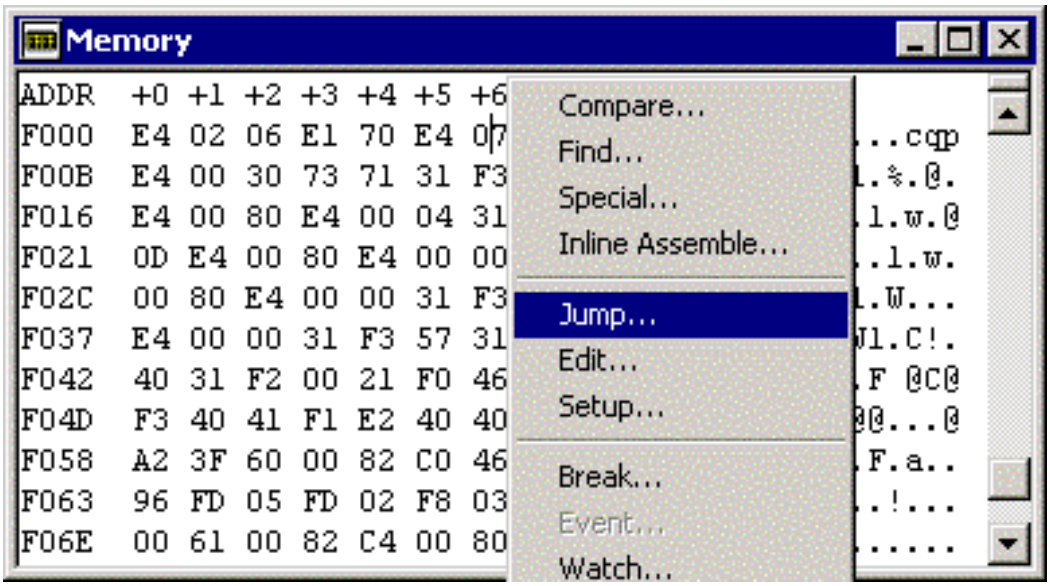
An example is given using the project made in Chapter 4 Tutorial.

《 Setting procedure 》

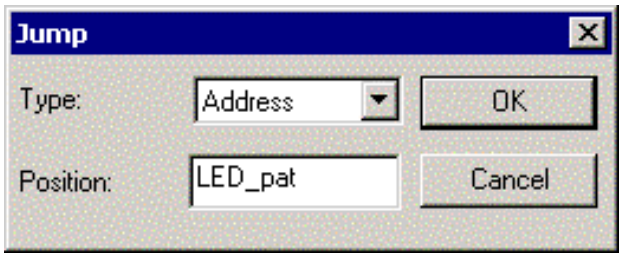
(1) Start the Simulator Debugger.

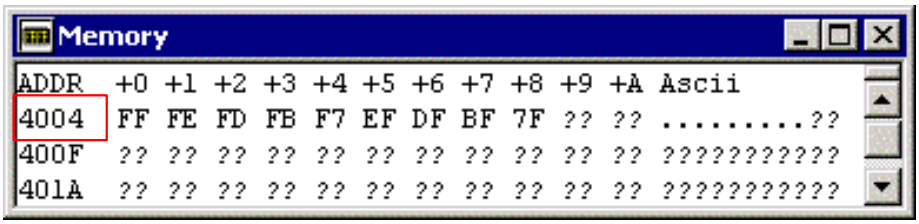
Select [Memory] from the [View] menu to open the Memory window.

Right-click the Memory window and select [Jump...] from the shortcut menu to open the [Jump] dialog.



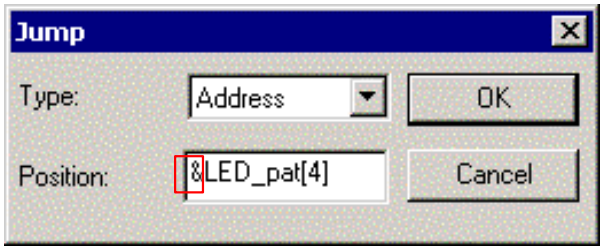
(2) When the array symbol “LED_pat” is specified for Position, a jump is made to address 4004 where the array is located.

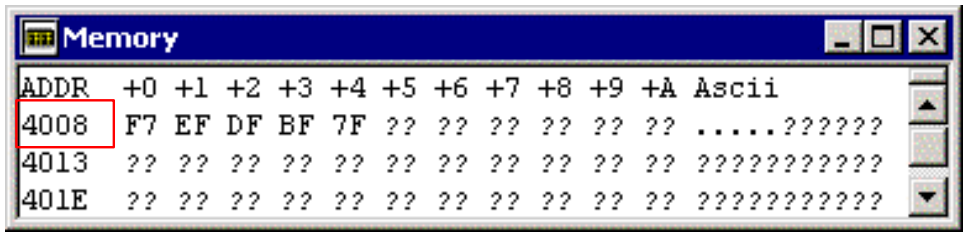




(3) Reopen the [Jump] dialog and specify the fourth array element.

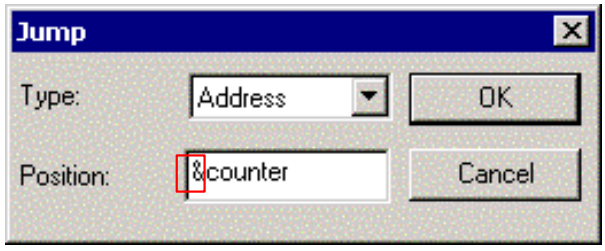
When “&” and the array element symbol “&LED_pat[4]” are specified for Position, a jump is made to address 4008 where the array element is located.

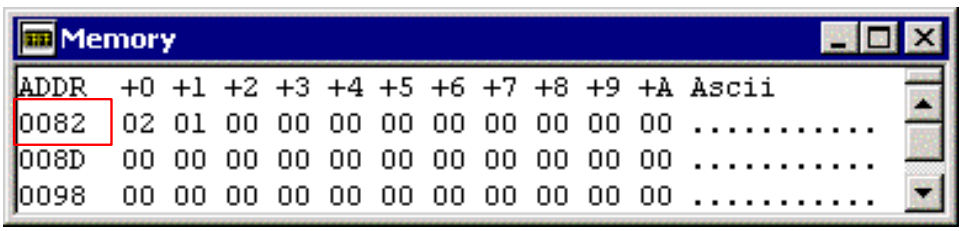




(4) Reopen the [Jump] dialog and specify the variable (counter).

When “&” and the variable symbol “&counter” are specified for Position, a jump is made to address 0082 where the variable is located.





Q3 How to set the path count of a breakpoint during using the Emulator Debugger

Answer

Help (***Section 4.6.4 Break Point*** of ***SOFTUNE Workbench Operation Manual***) describes how to set path count of a breakpoint. This function is enabled only in the Simulator mode.

The Emulator mode does not offer this function.

Q4 Why are the settings of breakpoints are changed when the target file is changed?

Answer

When the target file is changed, the settings of breakpoints and events specified by symbols are not changed unless the symbol names remain unchanged.

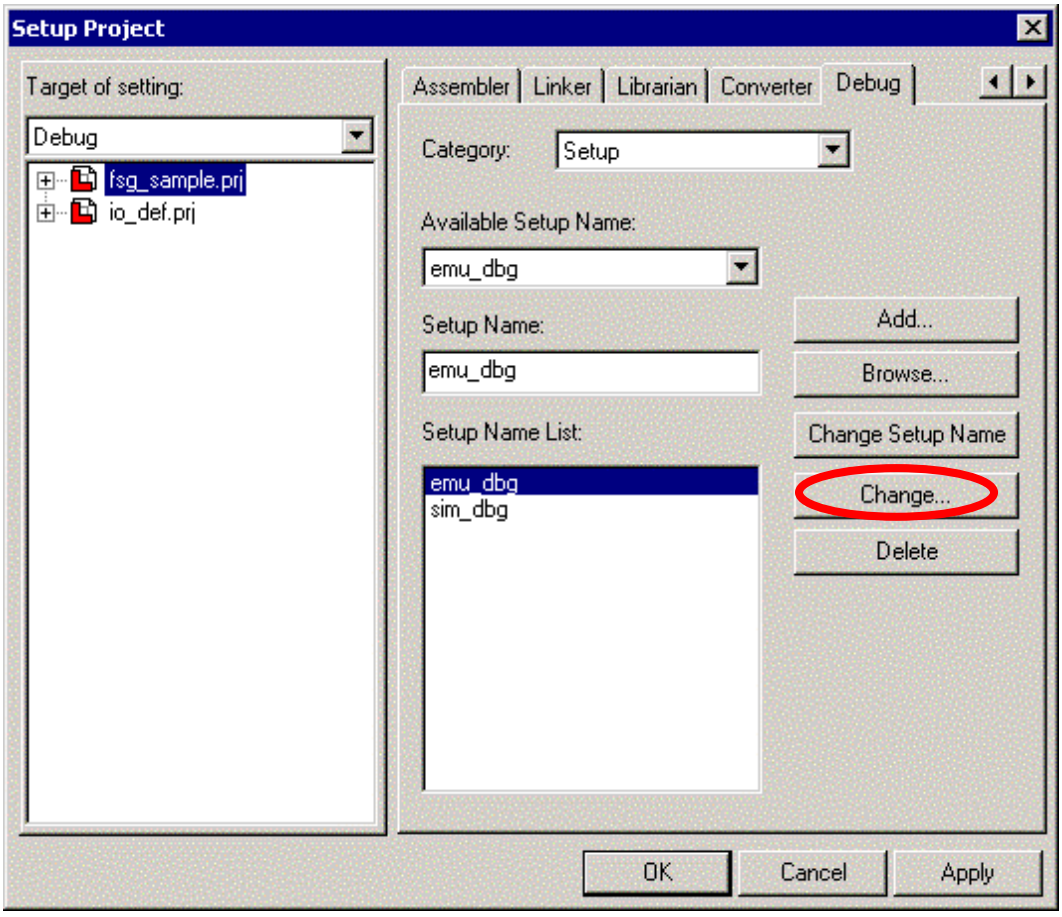
However, the settings of breakpoints and events specified at the absolute address must be reviewed if the source file is corrected to remake a target file.

An example of a setting from which setup breakpoint information is not inherited when loading the next target file is given below.

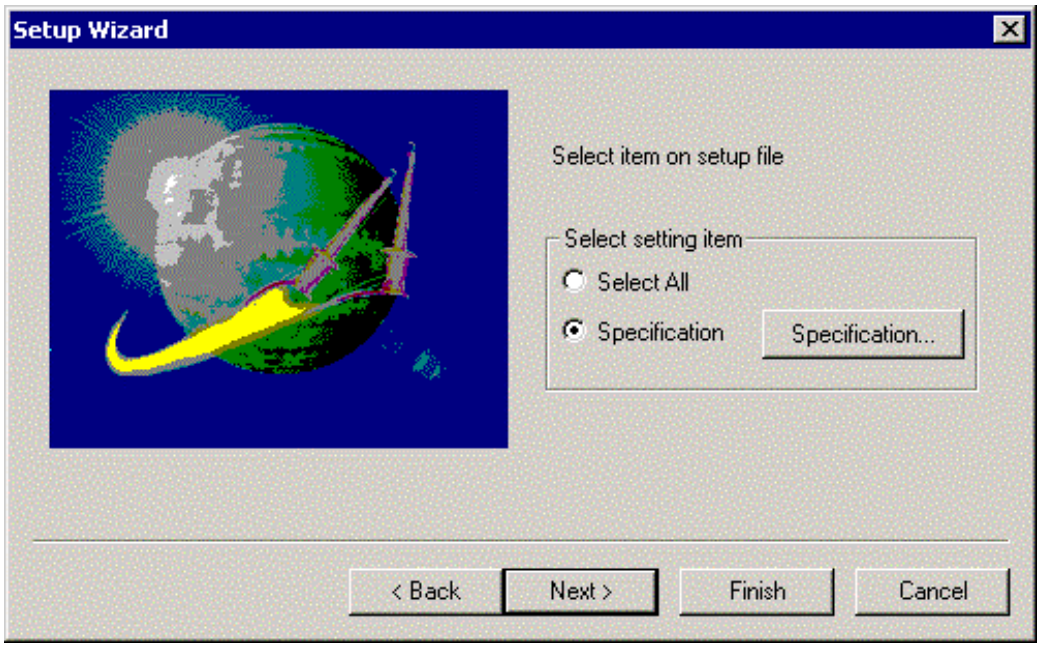
《 Setting procedure 》

(1) Select [Setup...] from the [Project] menu to open the [Setup Project] dialog.

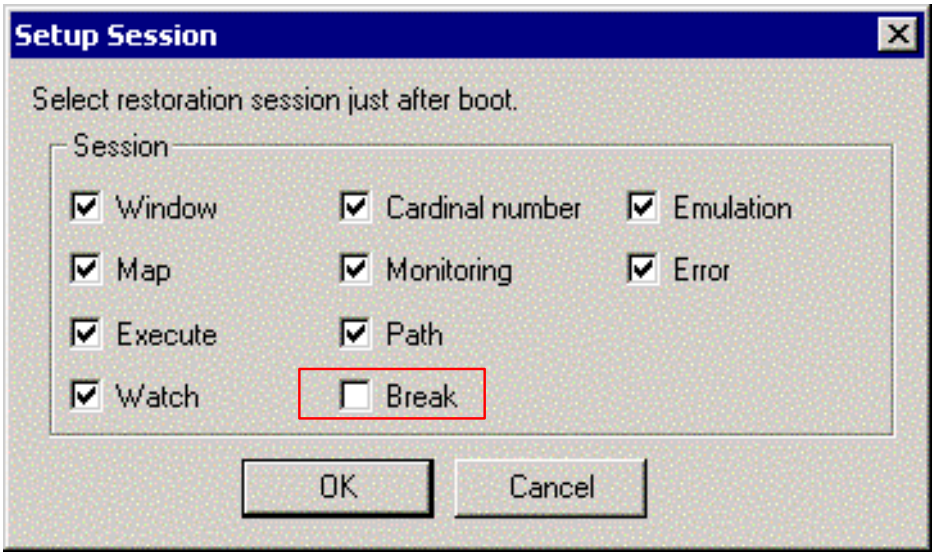
Select the setup file at Category in the [Debug] tab in the [Setup] dialog and click the [Change...] button.



(2) Follow the Wizard window, select “Specification” at the Select setting item and click the [Specification...] button.



(3) Uncheck the checkbox for “Break” in the [Setup Session] dialog.



Q5 How to save trace results and evaluation results such as memory dump to a file

Answer

To save the trace results as a file, right-click the Trace window and select [Save...] from the shortcut menu. Specify the file name.

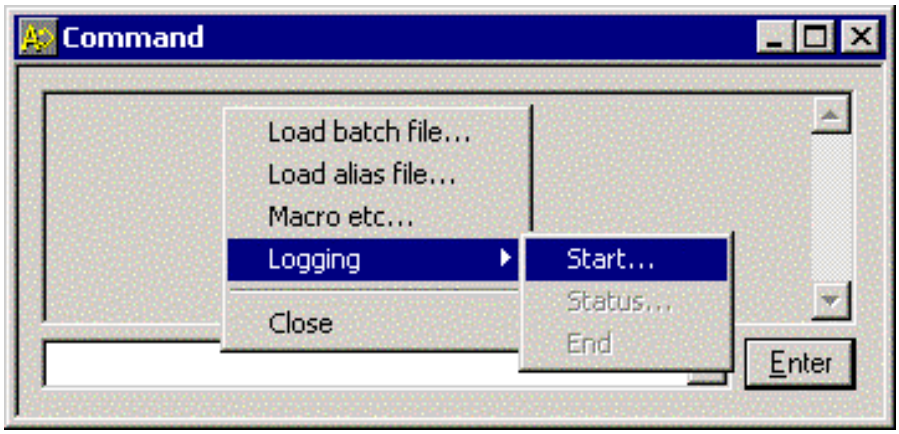
Information in the Output and Command windows can be logged to a file.

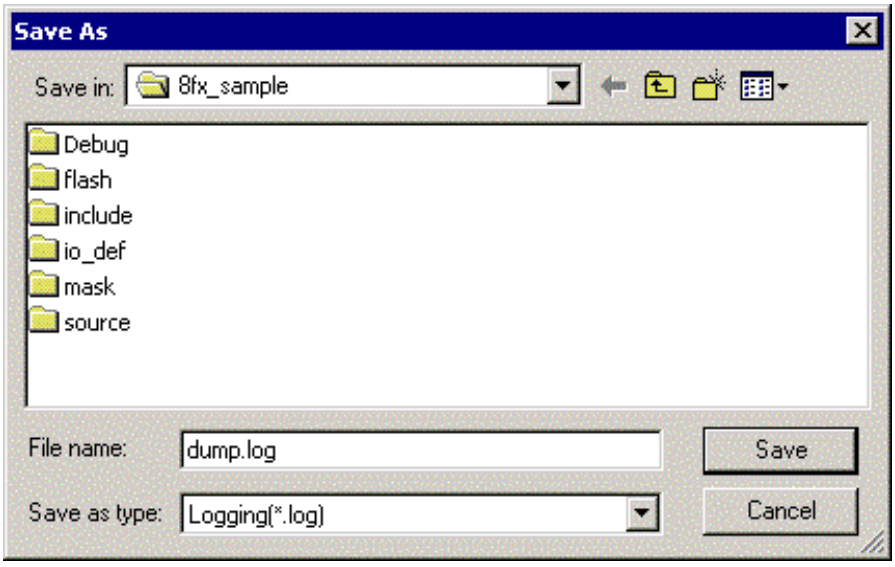
The procedure for logging information in the Command window to a file is given below.

《 Setting procedure 》

(1) Select [Command] from the [View] dialog to open the Command window.

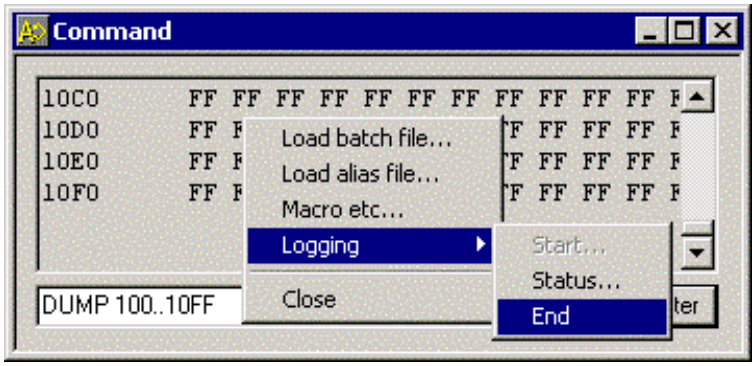
Right-click the Command window and select [Logging]-[Start...] from the shortcut menu. Specify the file save destination.

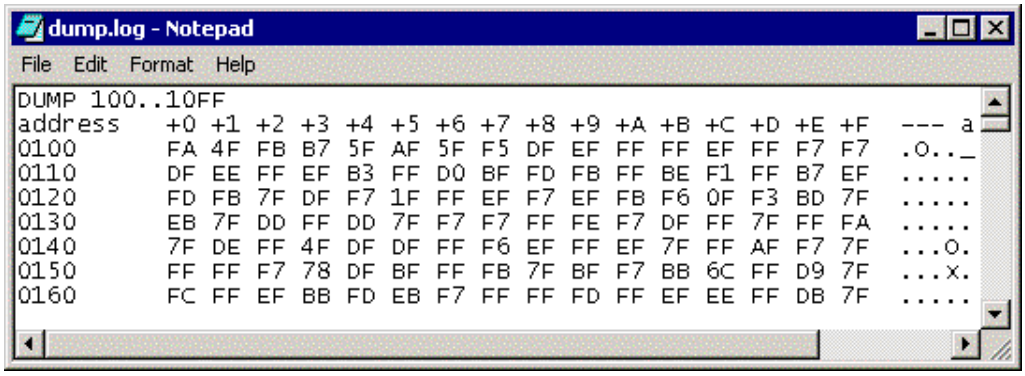




(2) Execute the “DUMP 100..10FF” command to display memory information on the Command window.

Right-click the Command window and select [Logging]-[End] from the shortcut menu. The log file is made in the specified folder.





Q6 Does the Emulator support coverage measurement?

Answer

Both the Emulator (MB2146-09) for the F^2MC-8FX and the Simulator do not support this function, so coverage measurement cannot be performed.

Q7 Does the Emulator support time measurement?

Answer

Although the Emulator (MB2141/A/B) for the F^2MC-8L has supported this function, the Emulator (MB2146-09) for the F^2MC-8FX does not. The Simulator supporting the function should be used to perform sufficient verification (debugging).

Q8 Does the Emulator support event setting?

Answer

Both the Emulator (MB2146-09) for the F^2MC-8FX and the Simulator do not support this function, so events cannot be set.

Q9 Does the Emulator support the Debugger memory map?

Answer

Although the Emulator (MB2141/A/B) for the F^2MC-8L has supported the memory map, the Emulator (MB2146-09) for the F^2MC-8FX does not, so guarded access cannot be detected. The Simulator supporting the memory map should be used to perform sufficient verification (debugging).

Q10 Does the Emulator support the monitoring function?

Answer

Although the Emulator (MB2141/A/B) for the F^2MC-8L has supported this function, the Emulator (MB2146-09) for the F^2MC-8FX does not. The Simulator supporting the function should be used to perform sufficient verification (debugging).

Q11 Are there any restrictions on the instruction count between branches that can be displayed by trace?

Answer

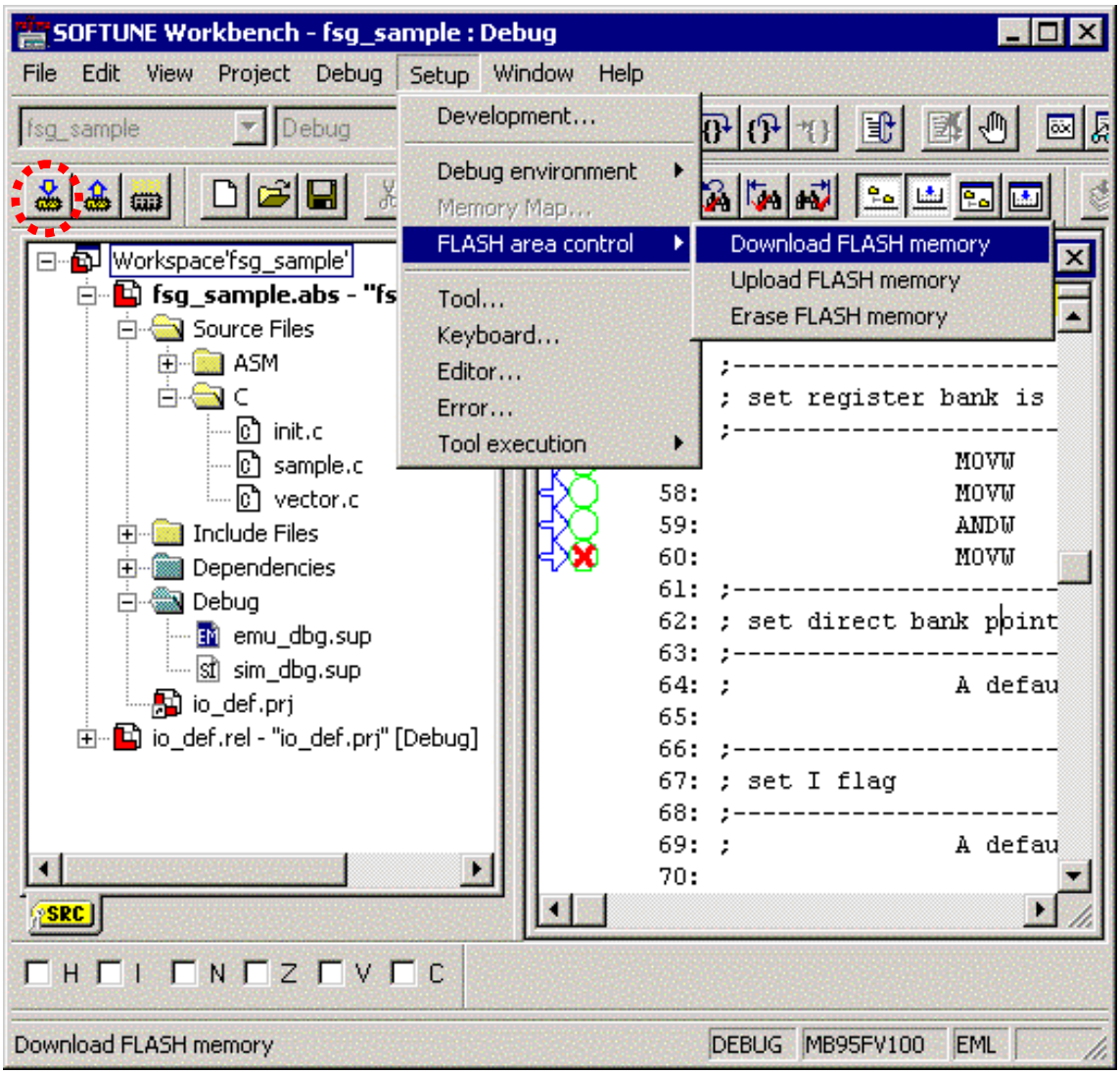
There are the following restrictions.

If 4094 or more branch instructions are not executed, only 4096 instructions from the branch address can be displayed.

Q12 What should I do if the progress bar showing “Processing” is displayed and the Debugger is not in the “execute” state when the program is executed by the Emulator?

Answer

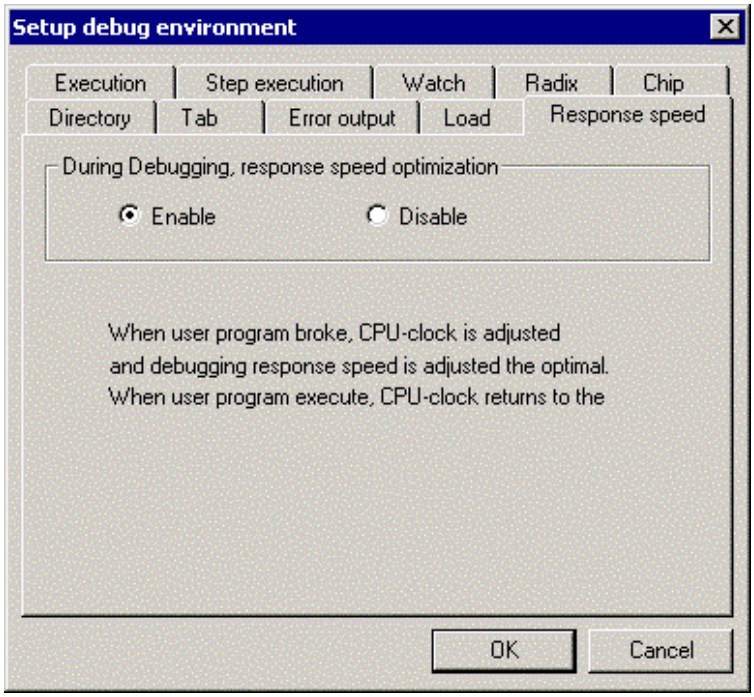
There is no problem. The Emulator (MB2146-09) for the F^2MC-8FX displays the progress bar when the program is executed after setting, changing, and deleting breakpoints or programming flash memory. When programming of flash memory is completed, the Debugger enters the “execute” state. “Processing” here means that programming of flash memory is in progress. To perform debugging in the “execute” state immediately after program execution, update flash memory ([Flash area control]-[Download Flash Memory]) or click the button (enclosed in dotted lines).



Q13 How to shorten the flash memory update time?

Answer

Select [Debug environment] from the [Setup] menu to open the [Setup debug environment] dialog. Set “Enable” for debugging response speed optimization in the [Response speed] tab. “Enable” is set in the Setup wizard for the Emulator Debugger by default.



Q14 What does the Debugger do if the user suspends the operation during executing in the standby mode by the Emulator?

Answer

When the user suspends the operation executing in the standby mode by the Emulator (MB2146-09) for the F^2MC-8FX, the Debugger breaks at the instruction next to the standby mode shift instruction.

C Others

Q1 How to check the version of *SOFTUNE* Workbench and each language tool.

Answer

The installation manual describes the version of the ***SOFTUNE*** tool.

Select the installation manual for each tool from [Program]-[Softune V3]-[Readme] in the Windows start menu and check each tool version.

[About xxxx] in the [Help] menu provides the versions of the following tools:

- ***SOFTUNE*** Workbench
- C Checker
- C Analyzer

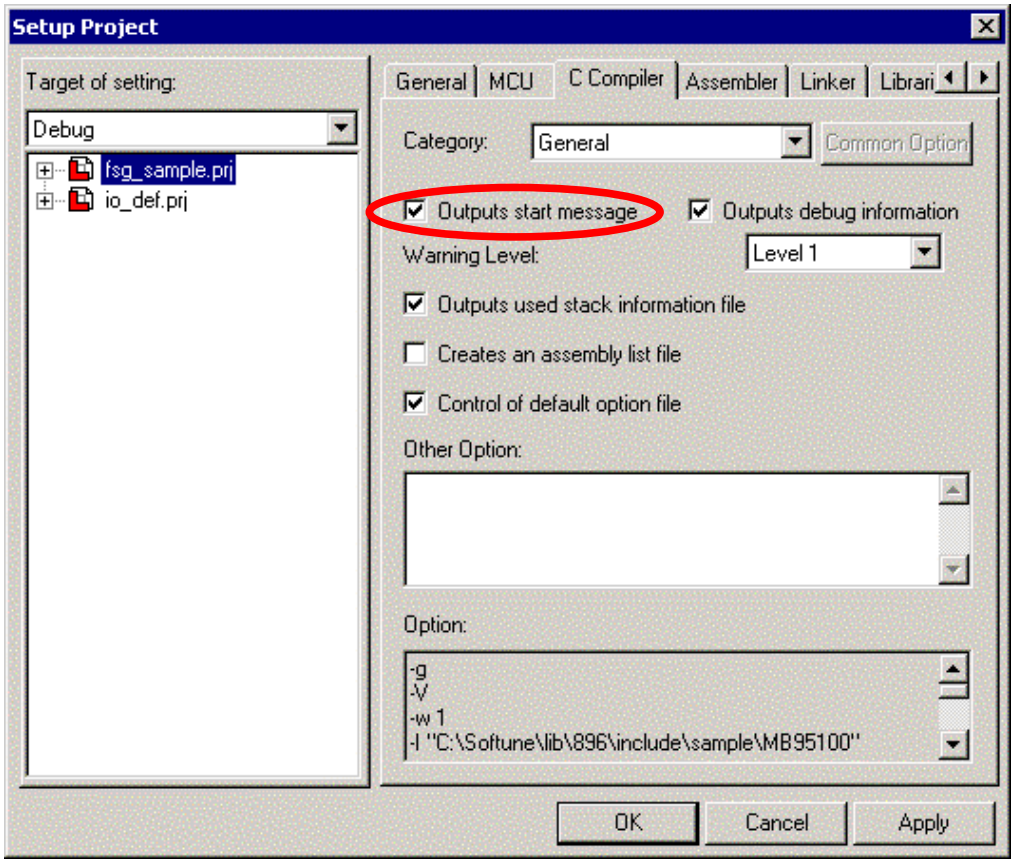
The user can check the version information on the F^2MC-8L/8FX Family ***SOFTUNE*** Professional Pack by the revision number on the CD-ROM or in the ***SOFTUNE Professional Pack Installation Manual*** (C:\Softune \ProPack896.txt).

Set the following items to display the version information on the language tools such as the Compiler and Assembler pack in the ***SOFTUNE*** Workbench Output window.

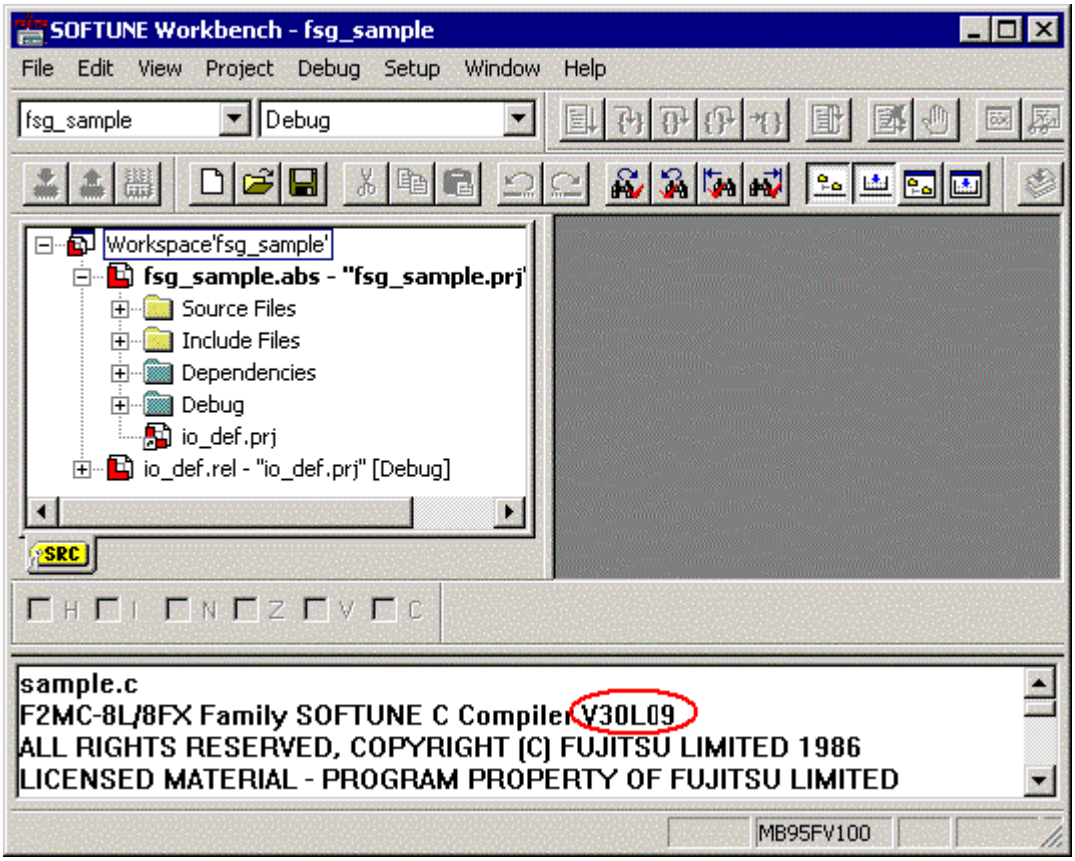
《 Setting procedure 》

(1) Select [Setup Project...] from the [Project] menu to open the [Setup Project] dialog.

Select the tab corresponding to the language tool version to be checked and put a checkmark in the “Outputs start message” checkbox.



(2) When [Compile], [Make], or [Build] is selected from the [Project] menu, the output message is output with version information in the Output window.



(**Note**: The version displayed in the window may be different from the product version.)

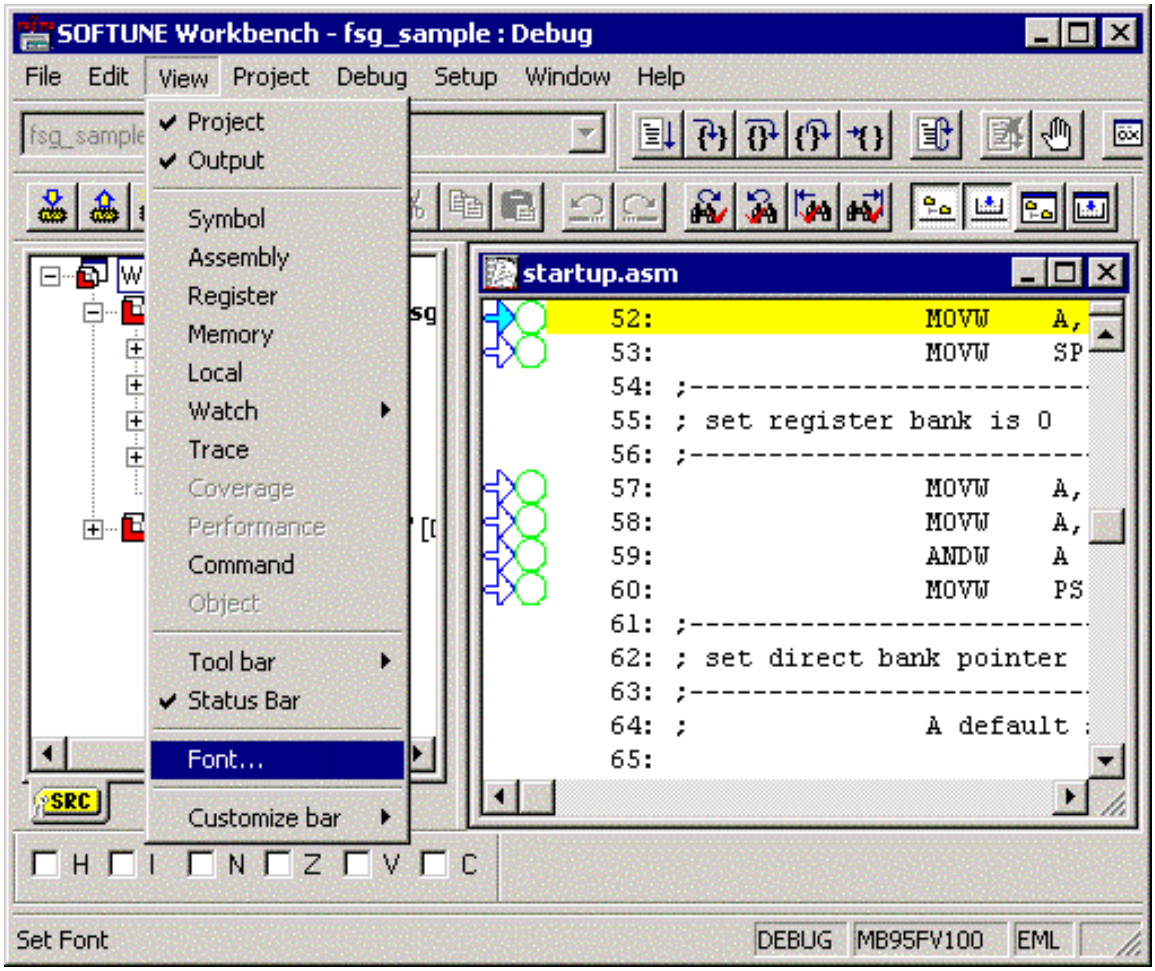
Q2 How to change the size of the font displayed in each *SOFTUNE* Workbench window.

Answer

The user can change the font size by the following procedure.

《 Setting procedure 》

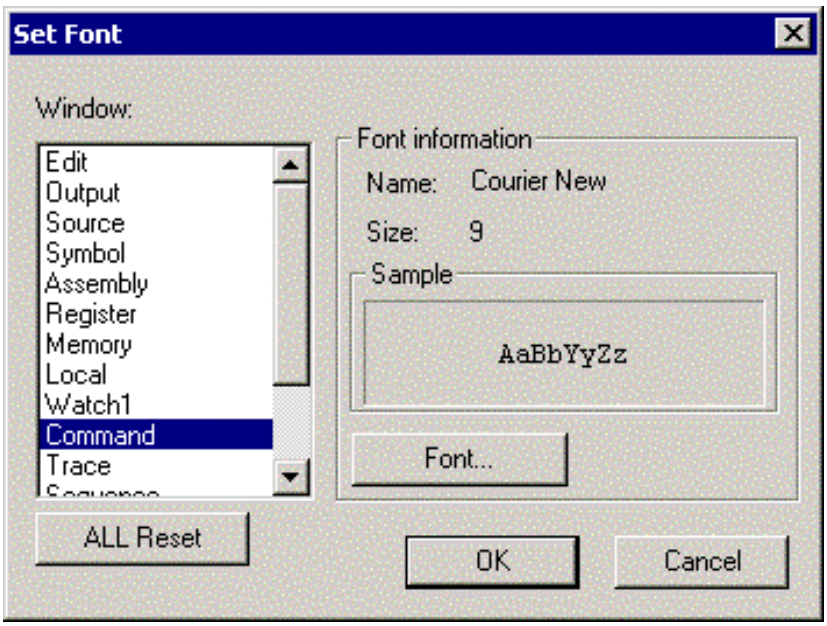
(1) Select [Font] from the [View] menu to open the [Set Font] dialog.

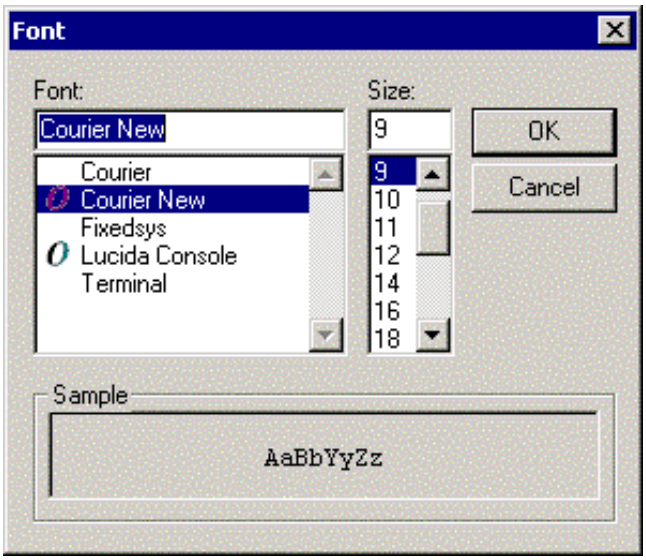


(2) Select the window to be changed and click the [Font...] button.

Change the font and size in the [Font] dialog.

The font and size can be changed in only the Edit window and Output window when not in the debug mode.





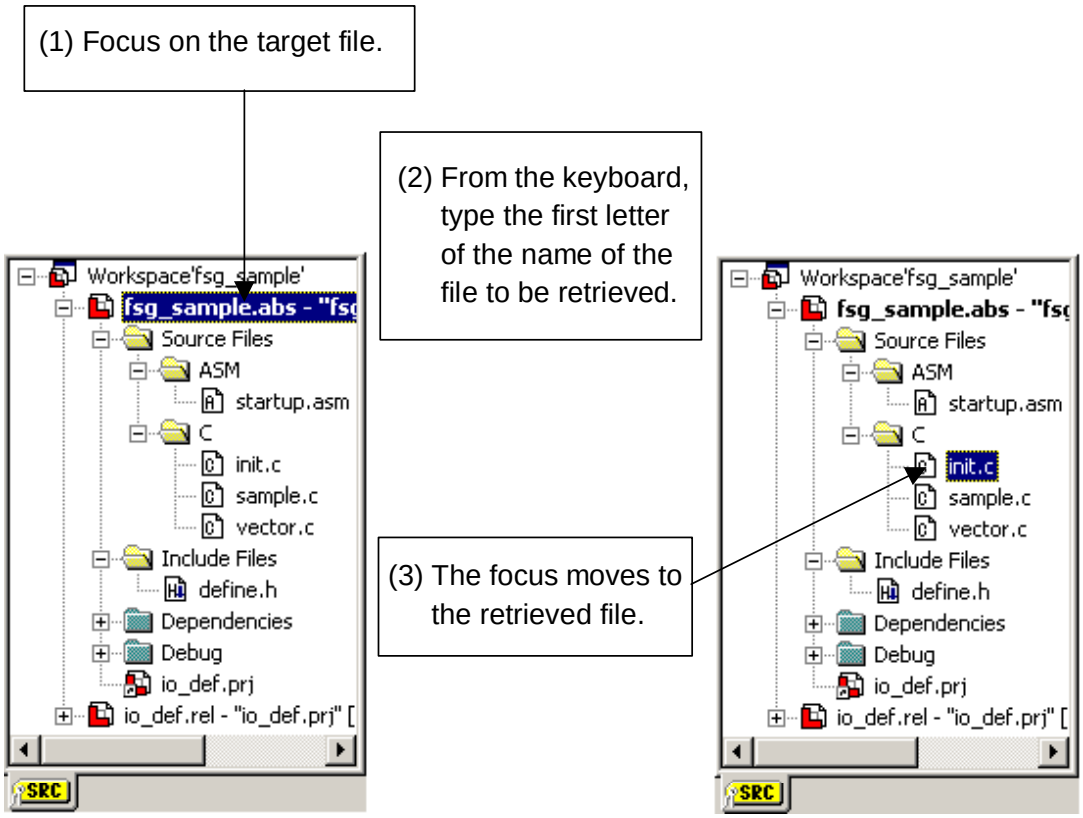
Q3 How to retrieve a file displayed in the *SOFTUNE* Workbench Project window.

Answer

When the first letter of the name of the file to be retrieved is input at the keyboard with focus anywhere in the Project window, the focus moves to the retrieved file.

Note that a file enclosed in a folder cannot be retrieved.

An example of retrieving is given below.



Q4 How to check details of the sample I/O register file.

Answer

The ***Sample I/O Register Files User's Manual*** describes the details of the sample I/O register file.

This ***Sample I/O Register Files User's Manual*** provides the manuals for describing the symbol defined by the sample I/O register file in the C source and for describing the symbol in the Assembler source in both Japanese and English versions as the following four text files for each product. Use the Editor to refer to the file for any purpose.

[Text file name]

ioregj.txt : Symbol description in C source (Japanese version)

ioreg.txt : Symbol description in C source (English version)

ioregj_a.txt : Symbol description in Assembler source (Japanese version)

ioreg_a.txt : Symbol description in Assembler source (English version)

[File directory]

MB95100 series : C:\Softune\lib\896\include\sample\MB95100

MB95110 series : C:\Softune\lib\896\include\sample\MB95110

MB95FV100 series : C:\Softune\lib\896\include\sample\MB95FV100

The user's manual for the symbol description in C source can also be opened using the following Windows start menu:

[Program] - [SoftuneV3] - [Sample IO Register Files]

- [FFMC-8L Family Sample IO Register Files USERS MANUAL]

- [MB95xxx series Sample IO Register Files USERS MANUAL]

Q5 How to change the record length of Motorola S format data.

Answer

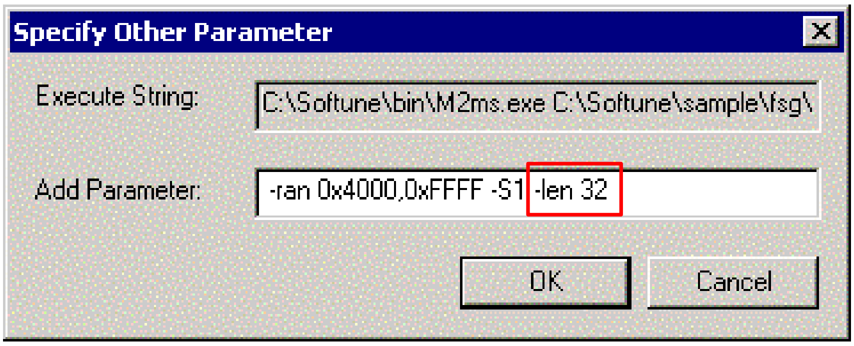
The default length of one record of Motorola S format data created by the load module Converter is fixed at 16 bytes.

To change the length, use the "-len" option for the S format Adjuster.

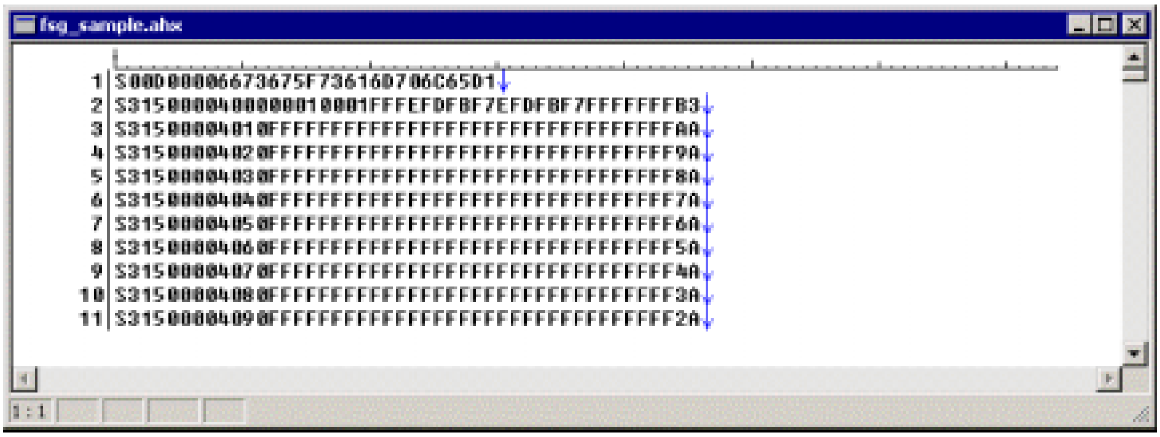
For the setting of the S format Adjuster, see ***Section 5.1 Setting External Tool***.

An example of the option to be added when the S format Adjuster is started is given below.

The example below changes, the length of one record of Motorola S format data to 32 bytes.



16 bytes



32 bytes

Q6 How to get the notices on using the Assembler in C.

Answer

The Accumulator can be used without restrictions but the general-purpose registers should be saved by the programmer.

For details, see Help (***Section 5.1 Assembler Description Function of SOFTUNE C Compiler***).

Q7 How to read projects developed by V30L26 or earlier *SOFTUNE* Workbench by V30L27 or later *SOFTUNE* Workbench?

Answer

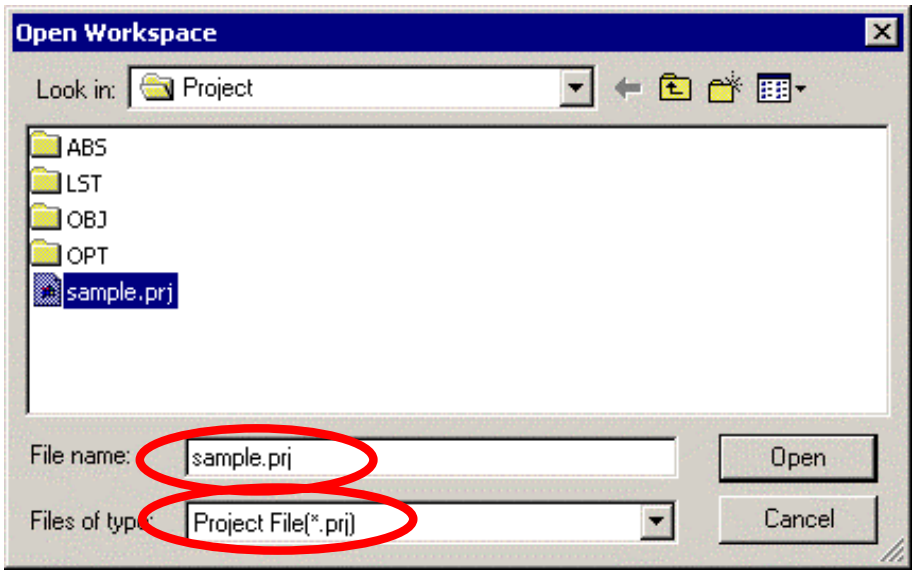
Projects developed by V30L26 or earlier versions of ***SOFTUNE*** Workbench have no workspace and have only project files. The project files can be read by V30L27 ***SOFTUNE*** Workbench.

The project Member files set previously and the settings of language tool options are all inherited.

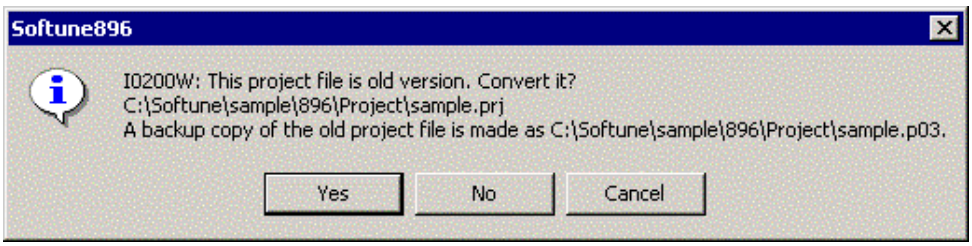
《 Setting procedure 》

(1) Select [Open Workspace] from the [File] menu to open the [Open Workspace] dialog.

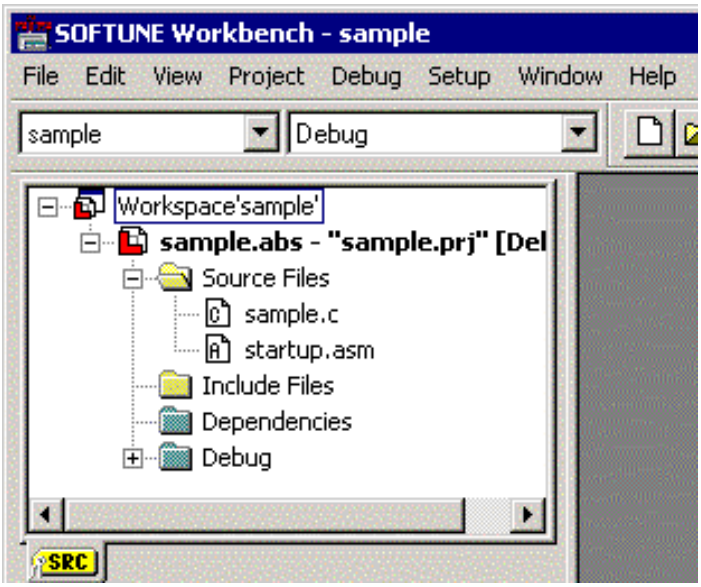
Specify Project File (~.**prj**) for Files of type and select the project file developed by V30L26 or earlier versions of ***SOFTUNE*** Workbench.



(2) When the dialog for confirming the conversion of the project file opens, click [Yes].



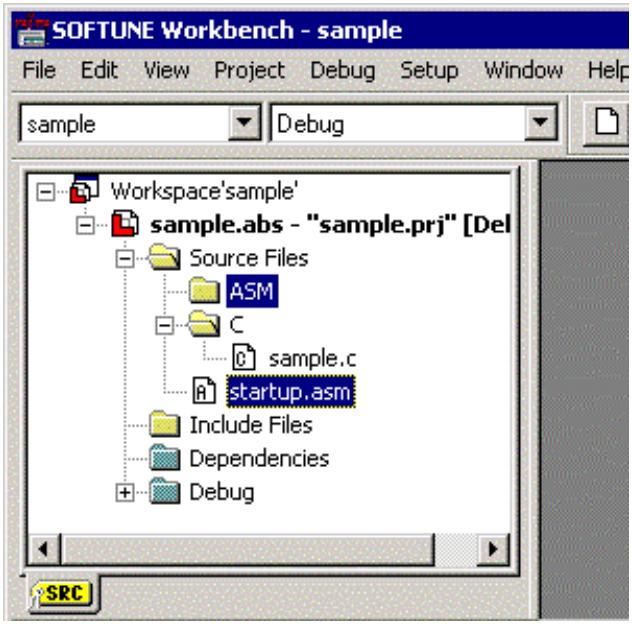
The inherited project Member information, dependencies, and settings of language tool options are made. When the workspace/project file is saved, a new workspace file will be made.



One Point !

= Sorting Member files =

All project Member files inherited from the previous version are stored in the Source Files folder. Create a folder as necessary to sort Member files. The drag-and-drop function facilitates moving of folders and Member files.



One Point !

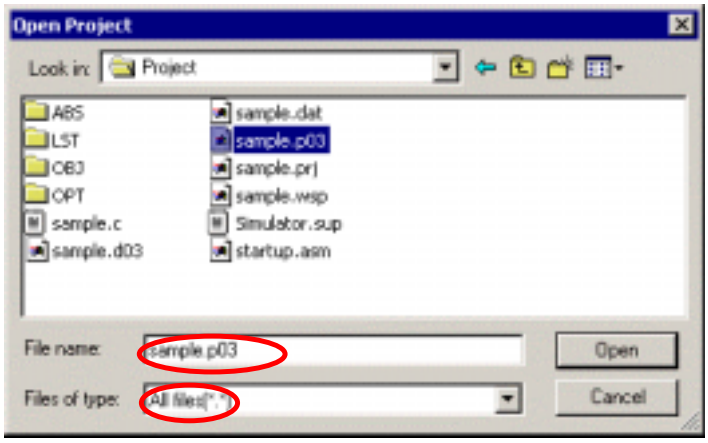
= Conversion and backup of project files =

Note that the project files read and saved by V30L27 or later versions of ***SOFTUNE*** Workbench cannot be opened by V30L26 or earlier versions of ***SOFTUNE*** Workbench.

To convert the project files, back up the project files and option data files under the following names. If the files already exist, they will not be overwritten.

- Project file project name.prj → project name.p03
- Option data file project name.dat → project name.d03

To reopen the project file using V30L26 or earlier version of ***SOFTUNE*** Workbench, read the backup file corresponding to this file.



F2MC-8FX FAMILY

SOFTUNE FIRST STEP GUIDE

MB2146-09 BGM Adapter

June 2004 **the first edition FUJITSU LIMITED**